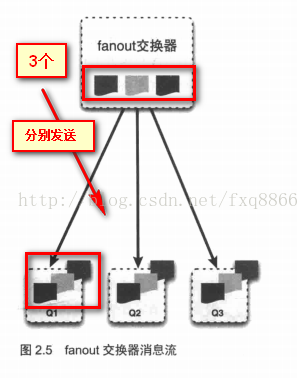
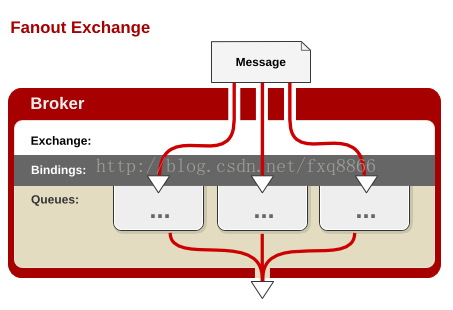
RabbitMQ服务器会根据路由键将消息从交换器路由到队列中，如何处理投递到多个队列的情况？这里不同类型的交换器起到了重要的作用。分别是fanout,direct，topic，每一种类型实现了不同的路由算法。

Fanout Exchange

  不处理路由键。你只需要简单的将队列绑定到交换机上。一个发送到交换机的消息都会被转发到与该交换机绑定的所有队列上。很像子网广播，每台子网内的主机都获得了一份复制的消息。Fanout交换机转发消息是最快的。



生产者

package com.dynamic.rabbitmy.ps;

/\*\*

\* Created by fxq on 2017/3/10.

\*/

import com.dynamic.rabbitmy.util.ConnectionUtil;

import com.rabbitmq.client.Channel;

import com.rabbitmq.client.Connection;

/\*\*

\* 生产者

\*/

public class Send {

private final static String EXCHANGE\_NAME="test\_exchange\_fanout";

public static void main(String[] args) throws Exception{

//获取到连接以及mq通道

Connection connection = ConnectionUtil.getConnection();

Channel channel = connection.createChannel();

//声明交换器

channel.exchangeDeclare(EXCHANGE\_NAME,"fanout");//fanout交换器

//消息内容

String message = "商品已经删除，id=1000";

channel.basicPublish(EXCHANGE\_NAME,"",null,message.getBytes());

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

channel.close();

connection.close();

}

}

消费者

package com.dynamic.rabbitmy.ps;

/\*\*

\* Created by fxq on 2017/3/10.

\*/

import com.dynamic.rabbitmy.util.ConnectionUtil;

import com.rabbitmq.client.Channel;

import com.rabbitmq.client.Connection;

import com.rabbitmq.client.QueueingConsumer;

/\*\*

\* 消费者

\*/

public class Recv

{

private final static String QUEUE\_NAME="test\_queue\_fanout\_1";

private final static String EXCHANGE\_NAME="test\_exchange\_fanout";

public static void main(String[] args) throws Exception{

//获取到连接以及通道

Connection connection = ConnectionUtil.getConnection();

Channel channel = connection.createChannel();

channel.queueDeclare(QUEUE\_NAME,false,false,false,null);

//绑定队列到交换器

channel.queueBind(QUEUE\_NAME,EXCHANGE\_NAME,""); //不设置路由键

//统一时刻服务器只会发一条消息给消费者;

channel.basicQos(1);

//定义队列的消费者

QueueingConsumer consumer = new QueueingConsumer(channel);

//监听队列，手动返回完成

channel.basicConsume(QUEUE\_NAME,false,consumer);

//获取消息

while (true)

{

QueueingConsumer.Delivery delivery = consumer.nextDelivery();

String message = new String(delivery.getBody());

System.out.println(" 前台系统：'" + message + "'");

Thread.sleep(10);

//手动返回

channel.basicAck(delivery.getEnvelope().getDeliveryTag(),false);

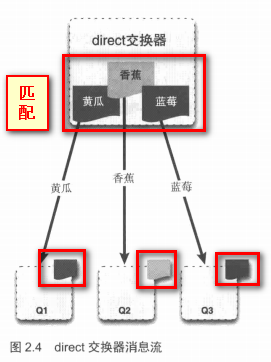
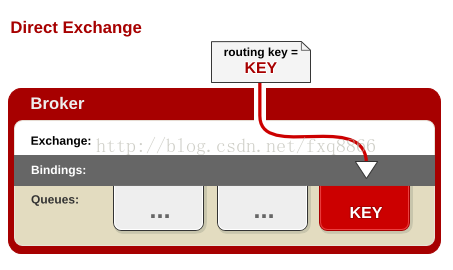
}

}

}

Direct Exchange

  处理路由键。需要将一个队列绑定到交换机上，要求该消息与一个特定的路由键完全匹配。这是一个完整的匹配。如果一个队列绑定到该交换机上要求路由键 “test”，则只有被标记为“test”的消息才被转发，不会转发test.aaa，也不会转发dog.123，只会转发test。



生产者：

package com.dynamic.rabbitmy.routing;

/\*\*

\* Created by fxq on 2017/3/10.

\*/

import com.dynamic.rabbitmy.util.ConnectionUtil;

import com.rabbitmq.client.Channel;

import com.rabbitmq.client.Connection;

/\*\*

\* 生产者

\*/

public class Send {

private final static String EXCHANGE\_NAME="test\_exchange\_direct";

public static void main(String[] args) throws Exception{

//获取到连接以及通道

Connection connection = ConnectionUtil.getConnection();

Channel channel = connection.createChannel();

//声明exchange

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

//消息内容

String message = "删除商品，id = 1001";

channel.basicPublish(EXCHANGE\_NAME,"delete",null,message.getBytes()); //此处delete为路由键；

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

channel.close();

connection.close();

}

}

生产者：

package com.dynamic.rabbitmy.routing;

/\*\*

\* Created by fxq on 2017/3/10.

\*/

import com.dynamic.rabbitmy.util.ConnectionUtil;

import com.rabbitmq.client.AMQP;

import com.rabbitmq.client.Channel;

import com.rabbitmq.client.Connection;

import com.rabbitmq.client.QueueingConsumer;

/\*\*

\* 消费者1

\*/

public class Recv {

private final static String QUEUE\_NAME="test\_queue\_direct\_1";

private final static String EXCHANGE\_NAME="test\_exchange\_direct";

public static void main(String[] args) throws Exception{

//获取连接以及mq通道

Connection connection = ConnectionUtil.getConnection();

Channel channel = connection.createChannel();

//声明队列

channel.queueDeclare(QUEUE\_NAME,false,false,false,null);

//绑定队列到交换机;

channel.queueBind(QUEUE\_NAME,EXCHANGE\_NAME,"update"); //匹配路由键为update

channel.queueBind(QUEUE\_NAME,EXCHANGE\_NAME,"delete"); //匹配路由键是delete

//同一时刻服务器只会发送一条消息给消费者;

channel.basicQos(1);

QueueingConsumer consumer = new QueueingConsumer(channel);

//监听队列，手动返回完成

channel.basicConsume(QUEUE\_NAME,false,consumer);

//获取消息

while (true)

{

QueueingConsumer.Delivery delivery = consumer.nextDelivery();

String message = new String(delivery.getBody());

System.out.println("前台系统：'"+message+"'");

Thread.sleep(10);

channel.basicAck(delivery.getEnvelope().getDeliveryTag(),false);

}

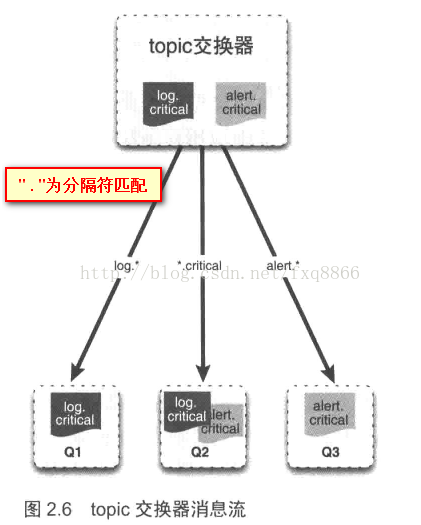
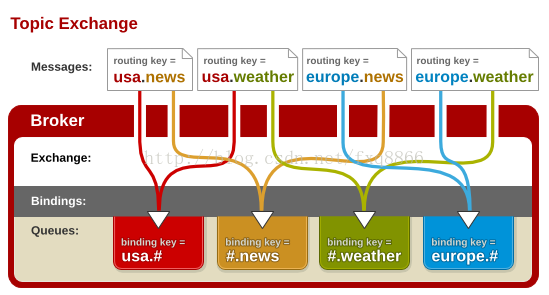
}

}

  该绑定在交换器上的队列，它可以匹配delete,update的路由键,但不是能匹配insert;必须和生产者声明是一模一样；

Topic Exchange

  将路由键和某模式进行匹配。此时队列需要绑定要一个模式上。符号“#”匹配一个或多个词，符号“\*”匹配不多不少一个词。因此“audit.#”能够匹配到“audit.irs.corporate”，但是“audit.\*” 只会匹配到“audit.irs”。



生产者：

package com.dynamic.rabbitmy.topic;

/\*\*

\* Created by fxq on 2017/3/10.

\*/

import com.dynamic.rabbitmy.util.ConnectionUtil;

import com.rabbitmq.client.Channel;

import com.rabbitmq.client.Connection;

/\*\*

\* 发送者

\*/

public class Send {

private final static String EXCHANGE\_NAME="test\_exchange\_topic" ;

public static void main(String[] args) throws Exception{

//获取到连接以及mq通道

Connection connection = ConnectionUtil.getConnection();

Channel channel = connection.createChannel();

//声明exchange

channel.exchangeDeclare(EXCHANGE\_NAME,"topic");

//消息内容

String message = "插入商品，id=100";

//发布消息

channel.basicPublish(EXCHANGE\_NAME,"item.insert",null,message.getBytes());

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

channel.close();

connection.close();

}

}

消费者：

package com.dynamic.rabbitmy.topic;

import com.dynamic.rabbitmy.util.ConnectionUtil;

import com.rabbitmq.client.Channel;

import com.rabbitmq.client.Connection;

import com.rabbitmq.client.QueueingConsumer;

import com.sun.media.sound.SF2InstrumentRegion;

/\*\*

\* Created by fxq on 2017/3/10.

\*/

public class Recv2 {

private final static String QUEUE\_NAME="test\_queue\_topic2";

private final static String EXCHANGE\_NAME="test\_exchange\_topic" ;

public static void main(String[] args) throws Exception{

//获得连接和mq通道

Connection connection = ConnectionUtil.getConnection();

Channel channel = connection.createChannel();

//声明通道

channel.queueDeclare(QUEUE\_NAME,false,false,false,null);

//绑定exchange

channel.queueBind(QUEUE\_NAME,EXCHANGE\_NAME,"item.#"); //使用item.# 匹配所有的以item开头的

//同一时刻服务器只能发送一条消息给消费者；

channel.basicQos(1);

//声明消费者

QueueingConsumer consumer = new QueueingConsumer(channel);

//监控队列，设置手动完成

channel.basicConsume(QUEUE\_NAME,false,consumer);

while (true)

{

QueueingConsumer.Delivery delivery = consumer.nextDelivery();

String message = new String(delivery.getBody());

System.out.println("搜索系统 '" + message + "'");

Thread.sleep(10);

channel.basicAck(delivery.getEnvelope().getDeliveryTag(),false);

}

}

}

  以上就是三种交换器的类型以及他们的使用场景，基于消息的路由键和交换器的类型，服务器会决定将消息投递到那个队列中。

————————————————

版权声明：本文为CSDN博主「随心更博」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/fxq8866/article/details/62049393