【RabbitMQ】三种类型交换器 Fanout,Direct,Topic

2017年03月14日 21:20:31 00润物无声00 阅读数：15312 标签： direct topic fanout RabbiMQ 更多

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。

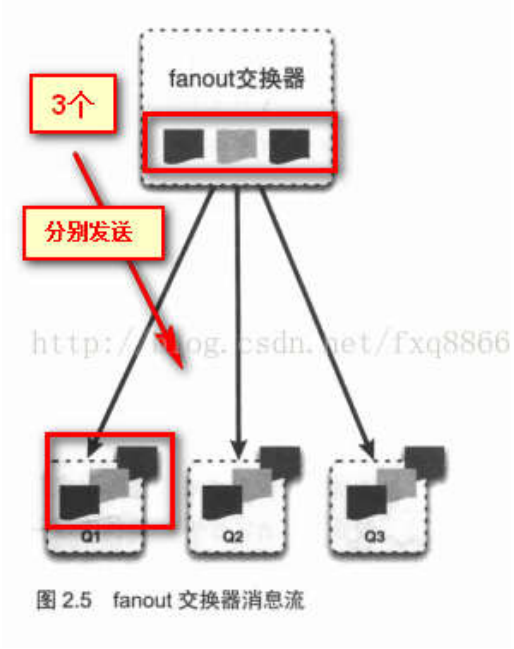
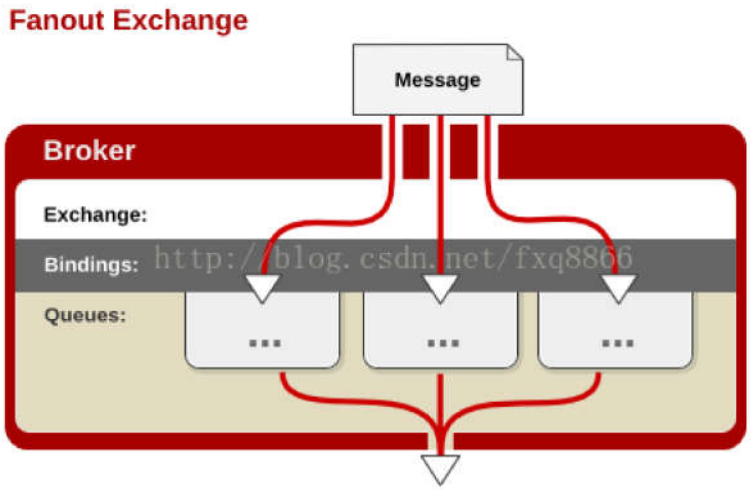


图 2.5 fanout 交换器消息流



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。

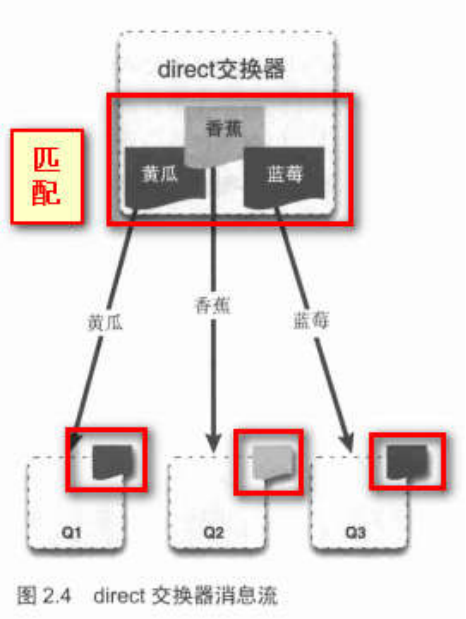
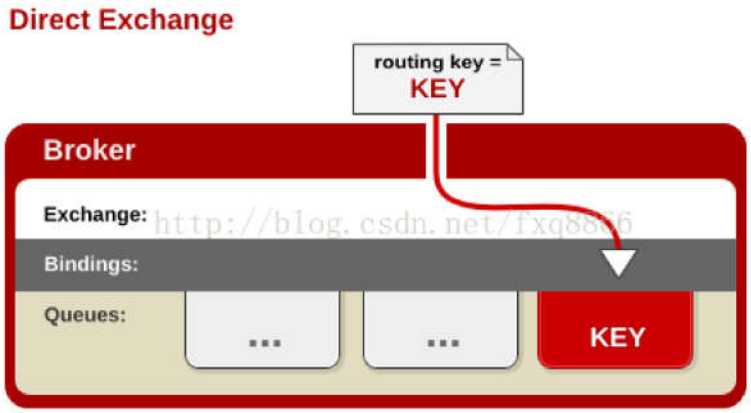


图 2.4 direct 交换器消息流



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”。

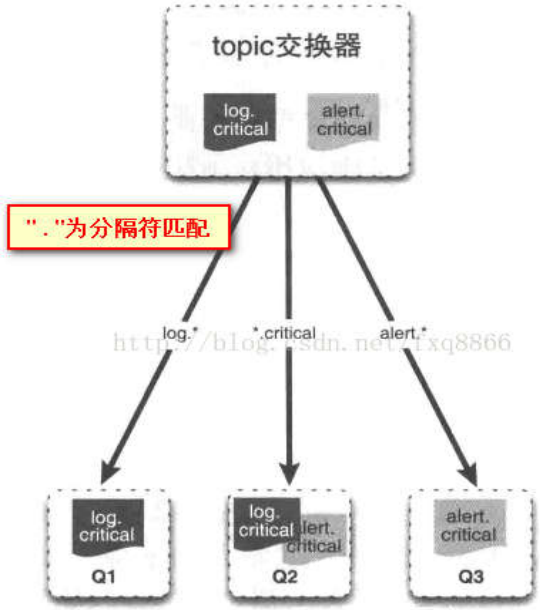
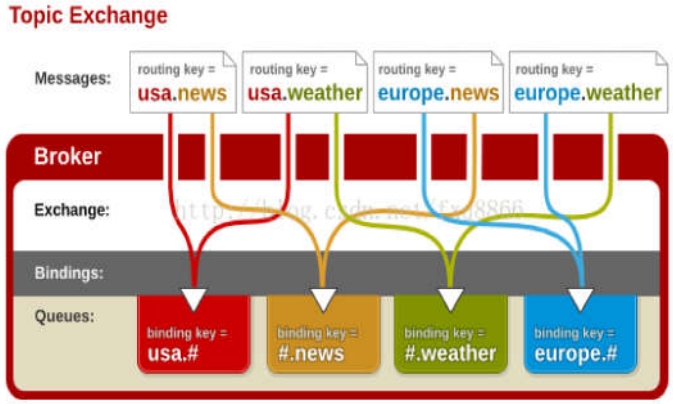


图 2.6 topic 交换器消息流



生产者：

```
1  package com.dynamic.rabbitmy.topic;
2
3  /**
4   * Created by fxq on 2017/3/10.
5   */
6
7  import com.dynamic.rabbitmy.util.ConnectionUtil;
8  import com.rabbitmq.client.Channel;
9  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 {13
    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。

想对作者说点什么

RabbitMQ三种Exchange模式(fanout,direct,topic) 4862

RabbitMQ三种Exchange模式(fanout,direct,topic)

RabbitMQ的Exchange 模式之direct （指定模式） 922

direct模式是rabbitmq的默认模式，看懂了下图基本就明白了该模式 在这个模式中，我们可以看到两个队列在direct... 来自： 浪丶荡

rabbitmq 交换模式-Direct 2311

direct 模式中 路由键 是必须的。import org.springframework.amqp.core.BindingBuilder; import org.springframewor... 来自： IT民工的博客

RabbitMQ Exchange 模式 - direct 4722

RabbitMQ 默认的 Exchange; 1. 消息传递时需要一个“RouteKey”，可以简单的理解为要发送到的队列名字。任何发送... 来自： 天外有天，人外有人。 专...

rabbitmq-direct（直接交换模式） 291

RabbitMQ—direct（直接交换模式） 一、什么是direct（直接交换模式） 生产者和消费者，具有相同的交换机名称（... 来自： fakerswe的博客

中间件系列五 RabbitMQ之Direct exchange（直连交换机）和路由 1420

概述上篇文章，我们通过Fanout exchange（扇型交换机）实现生产者发送一个消息，这个消息同时被传送给所有队... 来自： 不积跬步无以至千里; 不积...

RabbitMQ实例（三）——使用direct方式，带有Routingkey方式 3751

RabbitMQ实例（三）——使用direct方式，带有Routingkey方式 本文主要介绍的是使用direct方式的Exchange来进行... 来自： 止水的专栏

rabbitmq direct、fanout、topic 三种Exchange java 代码比较 2818

Producer端 1、channel的创建 无论是才用什么样的Exchange，创建channel代码都是相同的，如下 1 Connecti... 来自： 晨曦之星

Rabbitmq direct 模式：保证一个队列只对应一个消费者 处理方案

1140

问题描述：当消费者端在断网恢复时，在web后台发现，一个队列对应了两个消费者。当出现两个消费者时，生产者... 来自： chenjian60665的博客

spring-boot | rabbitMq-Direct模式

519

RabbitMQ是流行的开源消息队列系统，用erlang语言开发。RabbitMQ是AMQP（高级消息队列协议）的标准实现。 ... 来自： 代码_搬运工

Queue与Topic区别

2.1万

队列（Queue）和主题（Topic）是JMS支持的两种消息传递模型： 1、点对点（point-to-point，简称PTP）Que... 来自： 猴子哥哥的博客

RabbitMQ三种Exchange模式(fanout,direct,topic)的性能比较

4920

RabbitMQ中，所有生产者提交的消息都由Exchange来接受，然后Exchange按照特定的策略转发到Queue进行存储 R... 来自： 【无与科比】

RabbitMQ Consumer获取消息的两种方式(poll,subscribe)解析

4万

Producer和Queue Consumer和Queue 长连接 来自： 云计算、分布式架构、K8...

ActiveMQ学习总结——（三）Topic主题模式示例 3531

和队列模式相似，分别编写生产者和订阅者。 生产者：package com.jms.topic;import javax.jms.Connection; import... 来自： j253507692的专栏

Spring Boot RabbitMQ 入门（三）之 Fanout交换器 1929

0.回顾Spring Boot RabbitMQ 入门（二） 环境搭建 上篇文章我们学习了以下几点： * 1.如何创建交换器 * 2.如何创建... 来自： 自由

RabbitMQ直接模式（Direct） 30

2.2 直接模式（Direct） 2.2.1 什么是Direct模式 我们需要将消息发给唯一一个节点时使用这种模式，这是最简单的一... 来自： aileitianshi的博客

【RabbitMQ】rabbitmq交换器direct类型 61

demo意图：本次展示的demo意在消费对应的不同类型的日志类型。 生产者： package mq.direct; import com.rabbi... 来自： crainnogao的博客

RabbitMQ原理学习-- Direct交换器类型 68

任何发送到Direct Exchange的消息都会被转发到RouteKey中指定的Queue。1.这种模式下不需要将Exchange进行... 来自： 井底之蛙

RabbitMQ 消息队列 - direct 模式分发消息 121

推荐阅读 https://blog.csdn.net/column/details/15500.html direct 模式 根据 Binding 指定的 Routing Key, ... 来自： ...

RabbitMQ(四) : Direct 交换器(发布与订阅完全匹配) 17

Direct 类型交换器指消息发送者通过交换器完全匹配路由键的方式将消息绑定到相应队列，然后传递给相应的接收... 来自： 半路凉亭

RabbitMQ Exchange Type（Direct/Fanout/Topic）理解测试 1952

Exchanges 目前有4 种类型为 Direct， Fanout， Topic， handers 。以下测试简单理解 Exchanges, Routing key 及 ... 来自： KK 笔记：专注数据

RabbitMQ三种Exchange模式(fanout,direct,topic)介绍 1.4万

简介： RabbitMQ中，所有生产者提交的消息都由Exchange来接受，然后Exchange按照特定的策略转发到Queue进... 来自： 秋香的博客

（二）RabbitMQ消息队列-RabbitMQ消息队列架构与基本概念

5568

没错我还是没有讲怎么安装和写一个HelloWord，不过快了，这一章我们先了解下RabbitMQ的基本概念。RabbitMQ...

来自： Super_RD的博客

下载 **spring集成rabbitMq(基于direct、topic和fanout模式)**

11-15

spring集成rabbitMq(基于direct、topic和fanout模式)，包括main方法，5种情景，一天总结运行

下载 **rabbitmq三种exchange**

05-17

rabbitmq三种exchange方式：direct，fanout，topic发送和接收演示程序， 这是java版本，基于rabbitmq 3.1.0

RabbitMQ交换机Direct类型工作原理和PHP样例代码

193

1 RabbitMQ交换机Direct类型工作原理 2 PHP使用样例 http://php.net/manual/fa/book.amqp.php，使用文档。 消费...

来自： CleverCode的博客

RabbitMQ-从基础到实战（4）— 消息的交换（下）

395

0.目录 RabbitMQ-从基础到实战（1）— Hello RabbitMQ RabbitMQ-从基础到实战（2）— 防止消息丢失 RabbitMQ-从...

来自： duzanuolu的专栏

RabbitMQ（四）：分发到多消费者

1.2万

这篇文章中，我们将创建一个日志系统，它包含两个部分：第一个部分是发出log（Producer），第二个部分接收到...

来自： u010233323的博客

第5篇 RabbitMQ集成SpringBoot实现Direct模式

2791

直接代码 项目结构 pom需要增加对RabbitM的支持 Pom文件如下 xml version="1.0" encoding="UTF-8"?> project xm...

来自： u010753907的博客

RabbitMQ学习之四:发布/订阅(direct方式)

4257

参考http://blog.csdn.net/lmj623565791/article/details/37620057和RabbitMQ官网,加之自己部分修改和实验,因是新...

来自： xubingchuan_blog的博客

RabbitMQ各种交换机类型Exchange Types介绍

2万

最新版本的RabbitMQ有四种交换机类型，分别是Direct exchange、Fanout exchange、Topic exchange、Headers e...

来自： 苗雨顺的专栏

rabbitmq direct类型中，生产者端无法接收应答

1252

研究了一下rabbitmq，做了一个测试，出现的问题，这里记录一下，以便下次查...

来自： jackyxwr的专栏

Spring Boot RabbitMQ 入门（四）之 Topic交换器

3223

0.回顾Spring Boot RabbitMQ 入门（三）之 Fanout交换器 上篇文章我们学习了以下几点： * 1.如何创建消息生产者 *...

来自： 自由

利用Spring与ActiveMQ整合发送、接收消息实例(Queue与Topic模式)

4695

利用Spring与ActiveMQ整合发送、接收消息实例，同时使用Queue与Topic两种模式。

来自： u012358328的专栏

消息队列RabbitMQ与Spring集成

5.5万

RabbitMQ简介 Spring集成RabbitMQ 1 maven配置 2 rabbmitmq配置文件 3 Spring配置 在Spring中使用RabbitMQ 1 ...

来自： 唐僧打怪兽

Spring整合RabbitMQ简单示例

2084

首先导入工程必须的依赖,基本的spring的包就不用说了,除了这些还需要导入一下2个,用于整合 org.springframework....

来自： belovehejian的博客

RabbitMQ原理学习-- Topic交换器类型

86

topic类型简单来说和direct类似，只不过是topic的routeKey是一个匹配规则，如果匹配规则满足会将消息广播到Ex...

来自： 井底之蛙

（3）、exchange的几种类型及RPC的实现

539

在上一篇里，我们讲了rabbitMQ的通信流程。接下来，通过代码来实现exchange的几种类型，看一下rabbitMQ是如...

来自： 黄自豪的笔记本

RabbitMQ系列-SpringBoot创建三种类型交换机

125

@Configuration @PropertySource(value = "classpath:application.properties") public class R... 来自： M_Joes_17的博客

下载 **RabbitMQ入门代码** 06-30

rabbitMQ的java入门代码 包括队列持久化、消息持久化、direct、fanout、topic等的基础测试代码

下载 **基于rabbitmq的topic 交换** 05-07

基于rabbitmq的topic 消息交换模式，弥补了direct exchange和fanout exchange的不足，增加了其灵活性。

RabbitMQ （四） 路由选择 (Routing)

4.1万

上一篇博客我们建立了一个简单的日志系统，我们能够广播日志消息给所有你的接收者，如果你不了解，请查看：R... 来自： Hongyang

RabbitMQ-主题模式Topic

3.7万

Topic Exchange 将路由和某个模式匹配 # 匹配一个或者多个 * 匹配一个 例如 Good.insert Good.delete Good.#都能... 来自： 了无牵挂者忘生，心有所...

（八）RabbitMQ消息队列-通过Topic主题模式分发消息

6236

前两章我们讲了RabbitMQ的direct模式和fanout模式，本章介绍topic主题模式的应用。如果对direct模式下通过routi... 来自： Super_RD的博客

rabbitmq-----Routing和topic模式

591

前几篇已经说了mq的基本用法，也分别使用了普通的java写法和spring boot基于注解的方法去实现了mq的接收消息... 来自： java线程池

RabbitMQ与KafKa区别

6430

在应用场景方面， RabbitMQ,遵循AMQP协议，由内在高并发的erlanng语言开发，用在实时的对可靠性要求比较高... 来自： 【无与科比】

【RabbitMQ】三种Exchange模式——订阅、路由、通配符模式

1.2万

前两篇博客介绍了两种队列模式，这篇博客介绍订阅、路由和通配符模式，之所以放在一起介绍，是因为这三种模式... 来自： 颗粒归仓

RabbitMQ处理类型分类

1345

RabbitMQ处理的情况可以大体上分为7种情况 1、简单的单向发送和接收 一个发送者(Productor)和一个接收者(Com... 来自： 有情怀的梦想家

RabbitMQ: 交换类型 1477

RabbitMQ for Windows: Exchange Types Posted by Derek Greer on March 28, 2012 This is the four... 来自： Fred Lee的程序人生

RabbitMQ四种Exchange类型之Topic (Java) 729

Topic类型的Exchange是要进行路由键匹配的。此时队列需要绑定要一个交换器上。并有如下规则： 符号“#”匹配一... 来自： 活着离开这世界

00润物无声00 关注

原创	粉丝	喜欢	评论
173	134	198	4474

等级： 博客 访问： 38万+

积分： 1万+ 排名： 1580

勋章：

康得新








最新文章

MAC 安装Brew

springboot yml 配置文件注入Map，List

java保留2位小数及BigDecimal使用

Opensearch架构及引擎原理

Date类型使用Calendar进行时间运算

博主专栏

RabbitMq

热度：23453　　5 篇

SpringCloud

热度：43453　　16 篇

归档

2018年8月　2篇

2018年6月　1篇

2018年3月　1篇