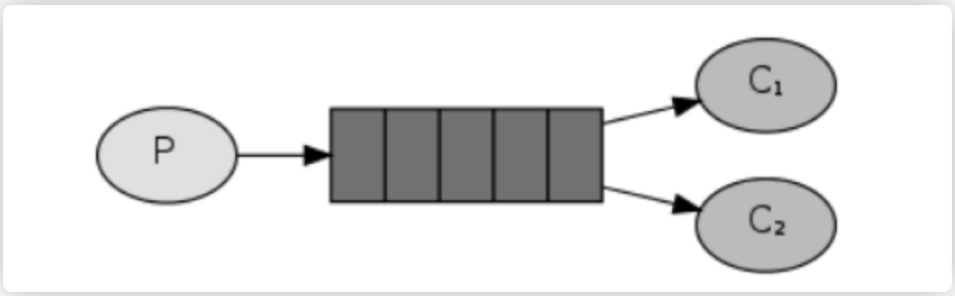
RabbitMQ系列（四）RabbitMQ進階-Queue隊列特性 (二)工作隊列 Work模式

原創 jzjie007 2020-06-20 22:22

work 工作隊列(工作模式)

工作模式就是work模式，1:n 指1個生產者 多個消費者，消費者存在競爭關係，只有一個消費者會獲得消息，進行消費，多個消費者競爭消費

比較適用於生產環境->負載均衡，能者多勞模式，如果機器網絡較好，處理速度較快，那麼採用這種方式，該機器消費消息就較多，可以通過basicQos來調整策略



工作模式上代碼

爲了區分效率，我們新建2個消費者，設置不同的消費延遲時間，另外把消息確認機制設置爲手動確認

2.1 生產者

依舊是Maven項目，項目 pom還是參考上一篇 新建maven工程，rabbitMQ簡單隊列

jzjie007

24小時熱門文章

來自某男的騷擾，沒想到可以給我帶來那麼大恐懼

放學後女兒哭着走出校門，ChatGPT寫了封信安慰她，卻又讓她紅了眼睛

230208《一年頂十年》7：通過講課輸出

快遞小哥與芯片女（117）

【柳梢青】早春

【塵世】中年危機與美麗的能力

中年辭職後關於再就業的一點思考

2.4 當上級兩面三刀，我們怎麼辦？

02-06 如何做到忙而不亂？

異性之間到底有沒有純潔的愛？

最新文章

RabbitMQ系列（三）RabbitMQ進階-Queue隊列特性 (一)簡單隊列

RabbitMQ系列（四）RabbitMQ進階-Queue隊列特性 (二)工作隊列 Work模式

RabbitMQ系列（一）啓動及網頁、外網訪問配置

RabbitMQ系列（二）VirtualHost作用及角色權限管理實戰

RabbitMQ系列-簡介

最新評論文章

linux以太網驅動總結

【python 圖片搜索】python 快速計算兩個圖片的相似度

```
package com.jzj.mq.second;

import com.jzj.mq.conn.MqConnectUtil;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;

/**
 * 當前描述:
 *
 * @author: jiazijie
 * @since: 2020/6/17 下午10:31
 */
public class SecondProducer {
    private final static String QUEUE_NAME = "queue_test";

    public static void main(String[] argv) throws Exception {
        // 獲取到連接以及mq通道
        Connection connection = MqConnectUtil.getConnectionDefault();
        // 從連接中創建通道
        Channel channel = connection.createChannel();

        /* 聲明（創建）隊列  queueDeclare( String queue, boolean durable, boolean exclusive,
boolean autoDelete,  Map<String, Object> arguments)
         * queue - 隊列名
         * durable - 是否是持久化隊列，  隊列的聲明默認是存放到內存中的，如果rabbitmq重啓會丟失
         * exclusie - 是否排外的，僅限於當前隊列使用
         * autoDelete - 是否自動刪除隊列，當最後一個消費者斷開連接之後隊列是否自動被刪除，可以通過界面
查看某個隊列的消費者數量，當consumers = 0時隊列就會自動刪除
         * arguments - 隊列攜帶的參數 比如 ttl-生命週期，x-dead-letter 死信隊列等等
         */
        channel.queueDeclare(QUEUE_NAME, false, false, false, null);

        //定義消息內容(發佈多條消息)
        for (int i = 0; i < 10; i++) {
            String message = "hello message " + i;
            /* 發送消息 String exchange, String routingKey, BasicProperties props, byte[] body
             * exchange - 交換機 , "" 空時候指定的是 獲取的virtualHost 虛擬服務器的 默認的exchang，每
個virtualHost都有一個AMQP default type:direct 直接轉發
             * queuename - 隊列信息
             * props - 參數信息
             * message 消息體 byte[]類型
             */
            channel.basicPublish("", QUEUE_NAME, null, message.getBytes());

            System.out.println(" **** Producer  Sent Message: '" + message + "'");

            //模擬發送消息延時，便於演示多個消費者競爭接受消息
            Thread.sleep(i * 10);
        }

        //關閉通道和連接
        channel.close();
        connection.close();
    }
}
```

2.2 消費者1和消費者2

消費者1

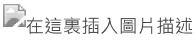
```
package com.jzj.mq.second;

import com.jzj.mq.conn.MqConnectUtil;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.QueueingConsumer;

/**
 * 當前描述:
 *
 * @author: jiazijie
 * @since: 2020/6/17 下午10:32
 */
public class SecondConsumer1 {
    private final static String QUEUE_NAME = "queue_test";

    public static void main(String[] argv) throws Exception {
        Connection connection = MqConnectUtil.getConnectionDefault();
        Channel channel = connection.createChannel();

        /*確保這裏的隊列是存在的*/
        channel.queueDeclare(QUEUE_NAME, false, false, false, null);

        //！！！！！ 同一時刻服務器只會發送一條消息給消費者
//        channel.basicQos(1);

        System.out.println(" **** Consumer->1 Waiting for messages. To exit press CTRL+C");


        QueueingConsumer consumer = new QueueingConsumer(channel);

        /* 消息確認機制
         * autoAck true:表示自動確認，只要消息從隊列中獲取，無論消費者獲取到消息後是否成功消費，都會認爲消息已經成功消費
         * autoAck false:表示手動確認，消費者獲取消息後，服務器會將該消息標記爲不可用狀態，等待消費者的反饋，如果消費者一直沒有反饋，那麼該消息將一直處於不可用狀態
         *          並且服務器會認爲該消費者已經掛掉，不會再給其發送消息，直到該消費者反饋
         *          !!!!!! 注意這裏是 false，手動確認
         */
        channel.basicConsume(QUEUE_NAME, false, consumer);

        while (true) {
            QueueingConsumer.Delivery delivery = consumer.nextDelivery();
            String message = new String(delivery.getBody());
            System.out.println(" **** Consumer->1 Received '" + message + "'");
            //消費者1 接收一條消息後休眠 100 毫秒，模仿複雜邏輯
             Thread.sleep(100);
            //返回確認狀態
            channel.basicAck(delivery.getEnvelope().getDeliveryTag(),false);
        }
    }
}
```

消費者2

```
package com.jzj.mq.second;

import com.jzj.mq.conn.MqConnectUtil;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.QueueingConsumer;

/**
 * 當前描述:
 *
 * @author: jiazijie
 * @since: 2020/6/17 下午10:32
 */
public class SecondConsumer2 {
    private final static String QUEUE_NAME = "queue_test";

    public static void main(String[] argv) throws Exception {
        Connection connection = MqConnectUtil.getConnectionDefault();
        Channel channel = connection.createChannel();

        /*確保這裏的隊列是存在的*/
        channel.queueDeclare(QUEUE_NAME, false, false, false, null);

        //！！！！！ 同一時刻服務器只會發送一條消息給消費者
//        channel.basicQos(1);

        System.out.println(" **** Consumer->2 Waiting for messages. To exit press CTRL+C");


        QueueingConsumer consumer = new QueueingConsumer(channel);

        /* 消息確認機制
         * autoAck true:表示自動確認，只要消息從隊列中獲取，無論消費者獲取到消息後是否成功消費，都會認爲消息已經成功消費
         * autoAck false:表示手動確認，消費者獲取消息後，服務器會將該消息標記爲不可用狀態，等待消費者的反饋，如果消費者一直沒有反饋，那麼該消息將一直處於不可用狀態
         *          並且服務器會認爲該消費者已經掛掉，不會再給其發送消息，直到該消費者反饋
         *          !!!!!! 注意這裏是 false，手動確認
         */
        channel.basicConsume(QUEUE_NAME, false, consumer);

        while (true) {
            QueueingConsumer.Delivery delivery = consumer.nextDelivery();
            String message = new String(delivery.getBody());
            System.out.println(" **** Consumer->2 Received '" + message + "'");
            //消費者1 接收一條消息後休眠 2000 毫秒，模仿複雜邏輯
             Thread.sleep(2000);
            //返回確認狀態
            channel.basicAck(delivery.getEnvelope().getDeliveryTag(),false);
        }
    }
}
```

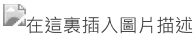
2.3 執行結果

先開啓消費者1，在開啓消費者2，然後再開啓生產者，生產消息



執行結果，可以看到，1條消息只能被1個消費者消費，不管消費耗時多久，都是平均消費？這就有點怪了，我牛逼，我還不能多幹點？很奇怪啊

!!!劃重點，mq消息的發送與消費與 basicQos有關，



2.4 basicQos 是什麼？

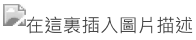
修改QOS後，查看執行結果

RabbitMQ提供了一種qos（服務質量保證）功能，即在非自動確認消息的前提下，如果一定數目的消息（通過基於consume或者channel設置Qos的值）未被確認前，不進行消費新的消息

👍 ɜsicQos 假如隊列中現在有100條消息，現在我們是手動確認消息機制，如果消費者消費能力差，那麼mq服
👎 器會一次性把所有消息都推送過來，我們一臺消費者客戶端，1分鐘同時要接收到300條消息，已經超過我們最大的負載，這時就可能導致，服務器資源被耗盡，消費者客戶端卡死等情況，basicQos就有作用了，用於流控

basicQos(10) 在手動確認的前提下，如果有10條消息沒有ack 應答，那麼我們就暫時不推送消息，如果有1條應答了，我們就推送1條(基本原理是這，中間還涉及1個消費者Prefetch count- 後面介紹作用)，保證系統穩定性，防止大量數據過來導致服務器掛掉

！！！現在我們把代碼中 basicQos 註釋放開，channel.basicQos(1); 再次測試一下功能



可以看到 能者多勞

消費者->1 休眠100ms，處理任務能力強，然後接收到了更多的MQ消息去消費
消費者->2 休眠2000ms，處理能力弱，只處理了個別消息
basicQos 等待消費者->1 處理完1條消息，恢復確認機制，當前消息已經被消費了，你可以下發下一條消息了，然後下發另一條，消費者1 再次接收到消息，再次消費，能者多勞

Java Web SprintBoot MQ中間件 rabbitmq 隊列 java spring

發表評論

登錄以後才評論...

登 录

所有評論

還沒有人評論，想成為第一個評論的人麼? 請在上方評論欄輸入並且點擊發布.

相關文章

springbootwebsocket

spring websocket簡單實踐 Spring提供的類和接口 spring 定義了一些接口和抽象類，我們只要實現這些接口和抽象類，就能夠完成一個目錄比較清晰的 WebSocket的服務器端，並且它裏面有很多封裝的功能，我們可以直接

原創 2023-02-02 01:32:31

Camunda簡介及開源協議

Camunda (Github，官網)是一個工作流引擎。目前提供二個主線版本7.X 與 8.X，二個主線版本架構上並不相同。 7.X與Activiti, Flowable 等開源工作流類似，以數據庫爲基礎，迭代完善的BPMN,DMN等功

原創 2023-02-01 23:36:06

O2OA(翱途)開發平臺服務器下載及安裝部署 For WindowsServer

O2OA(翱途)開發平臺[下稱O2OA開發平臺或者O2OA]支持公有云，私有云和混合雲部署，也支持複雜的網絡結構下的分佈式部署。O2OA(翱途)開發平臺安裝部署非常方便，只需要簡單的三步即可完成安裝。平臺內部集成了多項管理命令，可以

原創 2023-02-09 22:34:19

Solon 的插件熱插拔管理機制（H-Spi）

插件熱插拔管理機制，簡稱：H-Spi。是框架提供的生產時用的另一種高級擴展方案。相對E-Spi，H-Spi 更側重隔離、熱插熱拔、及管理性。 應用時，是以一個業務模塊爲單位進行開發，且封裝爲一個獨立插件包。 1、特點說明所有插件包獨享C

原創 2023-02-08 21:13:43

R2M分佈式鎖原理及實踐

作者：京東科技 張石磊 1 案例引入 名詞簡介： 資源：可以理解爲一條內容，或者圖+文字+鏈接的載體。 檔位ID: 資源的分類組，資源必須歸屬於檔位。 問題描述：當同一個檔位下2條資源同時審批通過時，收到擎天審批系統2條消息，消費者應用部

原創 2023-02-07 23:35:19

Java單元測試技巧之 JSON序列化

前言 《論語》中孔子有言：”工欲善其事，必先利其器。” 今年7月，作者希望迎接更大的挑戰，從高德地圖數據轉崗到共享出行後，接手並維護了幾個Java後端項目。在熟悉業務和代碼的過程中，快速地對原有項目進行單元測試用例的補充，使其單元測試覆蓋

原創 2023-02-06 12:41:30

那些年，我們寫過的無效單元測試

前言 那些年，爲了學分，我們學會了 面向過程編程； 那些年，爲了就業，我們學會了 面向對象編程； 那些年，爲了生活，我們學會了 面向工資編程； 那些年，爲了升職加薪，我們學會了 面向領導編程；

原創 2023-02-06 11:59:24

Java編程技巧之單元測試用例簡化方法

前言 清代譴責小說家吳趼人在《痛史》中寫道：“卷帙浩繁，望而生畏。” 意思是：“一部書的篇幅太長，讓人看見就害怕。”編寫單元測試用例也是如此，如果單元測試用例寫起來又長又複雜，自然而然地會讓人“望而生畏”，於是開始反感甚至於最終放棄。爲

原創 2023-02-04 00:00:52

我的新書《高性能Java系統權威指南》

各大書店正在銷售，停止內卷，學點實用的 本書代碼最新地址 https://gitee.com/xiandafu/java-performance 每章的readme.md 包含了主要代碼的說明 節選 書中章節 第1章 Java代

原創 2023-02-03 01:28:34

springwebsocket簡單demo

基於註解的websocket WebSocket 通信流程 ws:// 開頭 Upgrade:websocket Connection:upgrade 狀態碼：101協議變更 websocket沒有同源限制。 1. 服務終端類: 用java

原創 2023-02-01 13:28:31

java: JPS incremental annotation processing is disabled

在setting compile中修改大小

原創 2023-01-19 01:17:57

Java泛型01：基礎知識

1. 泛型程序設計 泛型是Java程序設計中一個重要的思想，它可以被用在類、接口、方法中。泛型簡單來說就是：1）所編寫的代碼在不用修改的前提下，可以被多種不同類型的對象所重用。 2）相較於雜亂的使用Object變量，泛型機制編寫的程序員

原創 2023-01-14 22:54:03

Java泛型02：泛型和虛擬機（類型擦除）

Java虛擬機（JVM，Java Virtual Machine）中並不存在泛型， Java 語言中的泛型只在程序源碼中存在，在編譯後的字節碼文件（Class 文件）中， 全部泛型都被替換爲原始類型，並且在相應的地方插入了 強制轉型代碼以及

原創 2023-01-14 22:54:02

記錄一次還算優雅的代碼設計

作者：京東零售 常文標 商卡聚合服務是一個小巧的rpc應用，功能是統一查詢商品的促銷、自營包郵、價格信息、區域庫存、區域可配送等等利益點或其他信息。本文重點分享商卡聚合服務的代碼設計，包括合理的Sirector線程調度（cpu使用率低），和

原創 2023-02-09 12:13:58

Redis實戰之session共享

當線上集羣時候，會出現session共享問題。 雖然Tomcat提供了session copy的功能，但是缺點比較明顯： 1：當Tomcat多的時候，session需要大量同步到多臺集羣上，佔用內網寬帶 2：同一個用戶session，需要在

原創 2023-02-06 22:52:33