链滴

RabbitMQ 通过 http API 获取队列数

作者：lonelyant
原文链接：https://ld246.com/article/1555466142517
来源网站：链滴

前提

通过API获取的前提是你的 rabbitmq_management处于开启状态，也就是能通过 http://host:15672 访问web管理端。RabbitMQ的安装与开启rabbitmq_management可以参考RabbitMQ安装、基础。

访问地址

上网找了一下发现没有几个说清楚了怎么直接用HTTP请求去获取队列数。其实完全不用去网上找的因为RabbitMQ自己就提供了HTTP API手册，比如我本地的API手册地址为：http://localhost:15672 api

获取队列详情API为

http://host:15672/api/queues/Vhost_name/queue_name

将上面的host换成RabbitMQ部署地址，Vhost_name换成队列所在的虚拟主机名，queue_name换队列名。也可以将Vhost_name、queue_name去掉通过http://host:15672/api/queues直接获取所队列信息。

访问结果是json串


```
⊟{
    "name":"Spider_Result_Queue_Test",
    "vhost":"Spider",
    "durable":true,
    "auto_delete":false,
    "exclusive":false,
    "arguments":⊞Object{...},
    "node":"rabbit@node4",
    "consumer_details":⊞Array[0],
    "deliveries":⊞Array[0],
    "incoming":⊞Array[0],
    "backing_queue_status":⊞Object{...},
    "disk_writes":0,
    "disk_reads":0,
    "head_message_timestamp":null,
    "message_bytes_persistent":0,
    "message_bytes_ram":0,
    "message_bytes_unacknowledged":0,
    "message_bytes_ready":0,
    "message_bytes":0,
    "messages_persistent":0,
    "messages_unacknowledged_ram":0,
    "messages_ready_ram":0,
    "messages_ram":0,
    "garbage_collection":⊞Object{...},
    "reductions":4922,
    "state":"running",
    "recoverable_slaves":null,
    "consumers":0,
    "exclusive_consumer_tag":null,
    "policy":null,
    "consumer_utilisation":null,
    "idle_since":"2018-11-16 9:38:02",
    "messages_unacknowledged_details":⊞Object{...},
    "messages_unacknowledged":0,
    "messages_ready_details":⊞Object{...},
    "messages_ready":0,
    "messages_details":⊞Object{...},
    "messages":0,
    "reductions_details":⊞Object{...},
    "message_stats":⊞Object{...},
    "memory":14096
}
```


原文链接：RabbitMQ 通过 http API 获取队列数

可以看到队列相关的所有信息都有记录。

注意

虚拟主机名Virtual host在设置的时候不要带/，不然会访问不到

{"error":"Object Not Found","reason":"\"Not Found\"\n"}

之前就是被这个坑了好久，明明按照API写的格式来的，就是访问不到。

代码

```
package com.ameya.utils;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import sun.misc.BASE64Encoder;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

/**
 * @author: Ant
 * @Date: 2018/11/23 13:28
 * @Description:
 */
@Component
public class MQUtils {
    private static final Logger logger = LoggerFactory.getLogger(MQUtils.class);

    @Value("${spring.rabbitmq.host}")
    private String host;
    @Value("${spring.rabbitmq.apiport}")
    private String port;
    @Value("${spring.rabbitmq.username}")
    private String username;
    @Value("${spring.rabbitmq.password}")
    private String password;
    @Value("${spring.rabbitmq.virtualHost}")
    private String virtualHost;

    /**
     * 队列任务总数
```

```
 *
 * @param queueName
 * @return
 */
public int getMessageCount(String queueName) throws IOException {
    String apiMessage = getApiMessage(queueName);
    if (Objects.equals(apiMessage, "")) {
        logger.error("请求RabbitMQ API时出错！！");
        return 0;
    }
    JSONObject jsonObject = JSON.parseObject(apiMessage);
    return Integer.parseInt(jsonObject.get("messages").toString());
}

/**
 * 队列ready任务数
 *
 * @param queueName
 * @return
 */
public int getMessageReadyCount(String queueName) throws IOException {
    String apiMessage = getApiMessage(queueName);
    if (Objects.equals(apiMessage, "")) {
        logger.error("请求RabbitMQ API时出错！！");
        return 0;
    }
    JSONObject jsonObject = JSON.parseObject(apiMessage);
    return Integer.parseInt(jsonObject.get("messages_ready").toString());
}

/**
 * 队列unack数MQ
 *
 * @param queueName
 * @return
 */
public int getMessagesUnacknowledgedCount(String queueName) throws IOException {
    String apiMessage = getApiMessage(queueName);
    if (Objects.equals(apiMessage, "")) {
        logger.error("请求RabbitMQ API时出错！！");
        return 0;
    }
    JSONObject jsonObject = JSON.parseObject(apiMessage);
    return Integer.parseInt(jsonObject.get("messages_unacknowledged").toString());
}

/**
 * 获取队列消息总数、ready消息数、unack消息数
 *
 * @param queueName
 * @return Map<String,Integer>
 */
public Map<String, Integer> getMQCountMap(String queueName) throws IOException {
    String apiMessage = getApiMessage(queueName);
```

```
        JSONObject jsonObject = JSON.parseObject(apiMessage);
        Map<String, Integer> map = new HashMap<>();
        map.put("messages", Integer.parseInt(jsonObject.get("messages").toString()));
        map.put("messages_ready", Integer.parseInt(jsonObject.get("messages_ready").toString()
);
        map.put("messages_unacknowledged", Integer.parseInt(jsonObject.get("messages_unac
nowledged").toString()));
        return map;
    }

    public String getApiMessage(String queueName) throws IOException {
        //发送一个GET请求
        HttpURLConnection httpConn = null;
        BufferedReader in = null;

        String urlString = "http://" + host + ":" + port + "/api/queues/" + virtualHost + "/" + qu
ueName;
        URL url = new URL(urlString);
        httpConn = (HttpURLConnection) url.openConnection();
        //设置用户名密码
        String auth = username + ":" + password;
        BASE64Encoder enc = new BASE64Encoder();
        String encoding = enc.encode(auth.getBytes());
        httpConn.setDoOutput(true);
        httpConn.setRequestProperty("Authorization", "Basic " + encoding);
        // 建立实际的连接
        httpConn.connect();
        //读取响应
        if (httpConn.getResponseCode() == HttpURLConnection.HTTP_OK) {
            StringBuilder content = new StringBuilder();
            String tempStr = "";
            in = new BufferedReader(new InputStreamReader(httpConn.getInputStream()));
            while ((tempStr = in.readLine()) != null) {
                content.append(tempStr);
            }
            in.close();
            httpConn.disconnect();
            return content.toString();
        } else {
            httpConn.disconnect();
            return "";
        }
    }
}
```