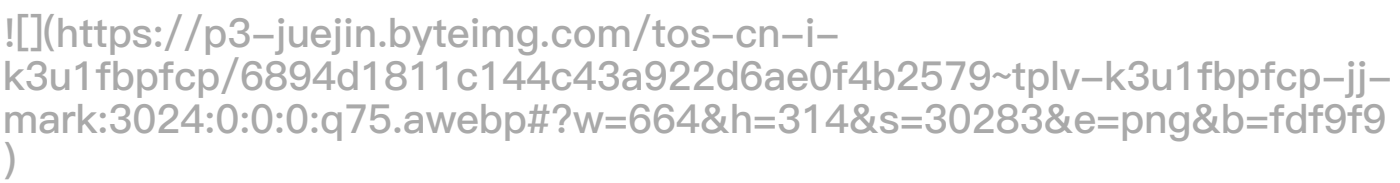
Redis stream 用做消息队列完美吗 ？

Redis Stream 是 Redis 5.0 版本中引入的一种新的数据结构，它用于实现简单但功能强大的消息传递模式。

这篇文章，我们聊聊 Redis Stream 基本用法 ， 以及如何在 SpringBoot 项目中应用 Redis Stream 。



1 基础知识

Redis Stream 的结构如下图所示，它是一个消息链表，将所有加入的消息都串起来，每个消息都有一个唯一的 ID 和对应的内容。

每个 Redis Stream 都有唯一的名称 ， 对应唯一的 Redis Key 。

同一个 Stream 可以挂载多个**消费组 ConsumerGroup** , 消费组不能自动创建，需要**使用 XGROUP CREATE 命令创建**。

每个消费组会有个**游标 last_delivered_id**，任意一个消费者读取了消息都会使游标 last_delivered_id 往前移动 ， 标识当前消费组消费到哪条消息了 。

消费组 ConsumerGroup 同样可以挂载多个消费者 Consumer , 每个 Consumer 并行的读取消息，任意一个消费者读取了消息都会使游标

last_delivered_id 往前移动。

消费者内部有一个属性 **pending_ids** ， 记录了当前消费者读取但没有回复 ACK 的消息 ID 列表 。

2 核心命令
======

01 XADD 向 Stream 末尾添加消息

使用 XADD 向队列添加消息，如果指定的队列不存在，则创建一个队列。基础语法格式：

```
XADD key ID field value [field value ...]
```

* **key** ： 队列名称，如果不存在就创建
* **ID** ： 消息 id，我们使用 * 表示由 redis 生成，可以自定义，但是要自己保证递增性。
* **field value** ： 记录。

```
127.0.0.1:6379> XADD mystream * name1 value1 name2 value2
”1712473185388-0”
127.0.0.1:6379> XLEN mystream
(integer) 1
127.0.0.1:6379> XADD mystream * name2 value2 name3 value3
”1712473231761-0”
```

消息 ID 使用 * 表示由 redis 生成，同时也可以自定义，但是自定义时要保证递增性。

> 消息 ID 的格式： 毫秒级时间戳 + 序号 , 例如：1712473185388-5 , 它表

示当前消息在毫秒时间戳 1712473185388 产生 ，并且该毫秒内产生到了第5条消息。

在添加队列消息时，也**可以指定队列的长度**。

```
127.0.0.1:6379> XADD mystream MAXLEN 100 * name value1 age 30
”1713082205042-0”
```

使用 XADD 命令向 `mystream` 的 stream 中添加了一条消息，并且指定了最大长度为 100。消息的 ID 由 Redis 自动生成，消息包含两个字段 `name` 和 `age`，分别对应的值是 `value1` 和 `30`。

02 XRANGE 获取消息列表

使用 XRANGE 获取消息列表，会自动过滤已经删除的消息。语法格式：

```
XRANGE key start end [COUNT count]
```

* **key** ： 队列名
* **start** ： 开始值， **-** 表示最小值
* **end** ： 结束值， **+** 表示最大值
* **count** ： 数量

```
127.0.0.1:6379> XRANGE mystream - + COUNT 2
1) 1) ”1712473185388-0”
   2) 1) ”name1”
      2) ”value1”
      3) ”name2”
      4) ”value2”
2) 1) ”1712473231761-0”
   2) 1) ”name2”
```

```
      2) "value2"
      3) "name3"
      4) "value3"

```

我们得到两条消息，第一层是消息 ID ， 第二层是消息内容 ， 消息内容是 Hash 数据结构 。

03 XREAD 以阻塞/非阻塞方式获取消息列表

使用 XREAD 以阻塞或非阻塞方式获取消息列表 ， 语法格式：

```
XREAD [COUNT count] [BLOCK milliseconds] STREAMS key [key ...] id [id ...]

```

* **count** ： 数量
* **milliseconds** ： 可选，阻塞毫秒数，没有设置就是非阻塞模式
* **key** ： 队列名
* **id** ： 消息 ID

```
127.0.0.1:6379> XREAD streams mystream 0-0
1) 1) "mystream"
   2) 1) 1) "1712473185388-0"
         2) 1) "name1"
            2) "value1"
            3) "name2"
            4) "value2"
      2) 1) "1712473231761-0"
         2) 1) "name2"
            2) "value2"
            3) "name3"
            4) "value3"

```

XRED 读消息时分为**阻塞**和**非阻塞**模式，使用 **BLOCK** 选项可以

表示阻塞模式，需要设置阻塞时长。非阻塞模式下，读取完毕（即使没有任何消息）立即返回，而在阻塞模式下，若读取不到内容，则阻塞等待。

```
127.0.0.1:6379> XREAD block 1000 streams mystream $
(nil)
(1.07s)
```

使用 Block 模式，配合 $ 作为 ID ，表示读取最新的消息，若没有消息，命令阻塞！等待过程中，其他客户端向队列追加消息，则会立即读取到。

因此，典型的队列就是 XADD 配合 XREAD Block 完成。XADD 负责生成消息，XREAD 负责消费消息。

04 XGROUP CREATE 创建消费者组

使用 XGROUP CREATE 创建消费者组，分两种情况：

* 从头开始消费:

```
XGROUP CREATE mystream consumer-group-name 0-0
```

* 从尾部开始消费:

```
XGROUP CREATE mystream consumer-group-name $
```

执行效果如下：

```
127.0.0.1:6379> XGROUP CREATE mystream mygroup 0-0
OK
```

05 XREADGROUP GROUP 读取消费组中的消息

使用 XREADGROUP GROUP 读取消费组中的消息，语法格式：

```
XREADGROUP GROUP group consumer [COUNT count] [BLOCK
milliseconds] [NOACK] STREAMS key [key ...] ID [ID ...]
```

* **group**： 消费组名
* **consumer**： 消费者名。
* **count**： 读取数量。
* **milliseconds**： 阻塞毫秒数。
* **key**： 队列名。
* **ID**： 消息 ID。

示例：

```
127.0.0.1:6379>  XREADGROUP group mygroup consumerA count 1
streams mystream >
1) 1) "mystream"
   2) 1) 1) "1712473185388-0"
         2) 1) "name1"
            2) "value1"
            3) "name2"
            4) "value2"
```

消费者组 `mygroup` 中的消费者 `consumerA` ，从 名为 `mystream` 的 Stream 中读取消息。

* `COUNT 1` 表示一次最多读取一条消息
* `>` 表示消息的起始位置是当前可用消息的 ID，即从当前未读取的最早消息开始读取。

06 XACK 消息消费确认

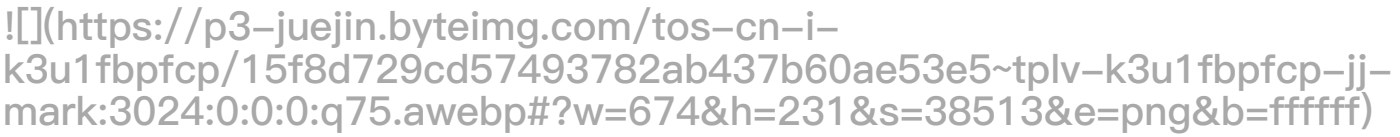
接收到消息之后，我们要手动确认一下（ack），语法格式：

```
xack key group-key ID [ID ...]
```

示例：

```
127.0.0.1:6379> XACK mystream mygroup 1713089061658-0
(integer) 1
```

消费确认增加了消息的可靠性，一般在业务处理完成之后，需要执行 ack 确认消息已经被消费完成，整个流程的执行如下图所示：



我们可以使用 xpending 命令查看**消费者未确认的消息ID**：

```
127.0.0.1:6379> xpending mystream mygroup
1) (integer) 1
2) "1713091227595-0"
3) "1713091227595-0"
4) 1) 1) "consumerA"
```

```
      2) "1"
```

07 XTRIM 限制 Stream 长度

我们使用 XTRIM 对流进行修剪，限制长度， 语法格式：

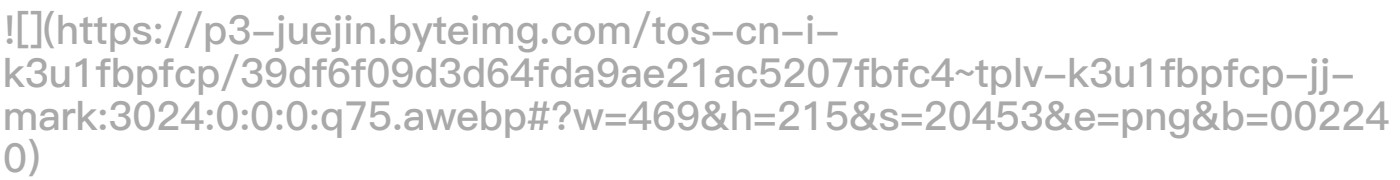
```
127.0.0.1:6379> XADD mystream * field1 A field2 B field3 C field4 D
"1712535017402-0"
127.0.0.1:6379> XTRIM mystream MAXLEN 2
(integer) 4
127.0.0.1:6379> XRANGE mystream - +
1) 1) "1712498239430-0"
   2) 1) "name"
      2) "zhangyogn"
2) 1) "1712535017402-0"
   2) 1) "field1"
      2) "A"
      3) "field2"
      4) "B"
      5) "field3"
      6) "C"
      7) "field4"
      8) "D"
```

3 SpringBoot Redis Stream 实战
==========

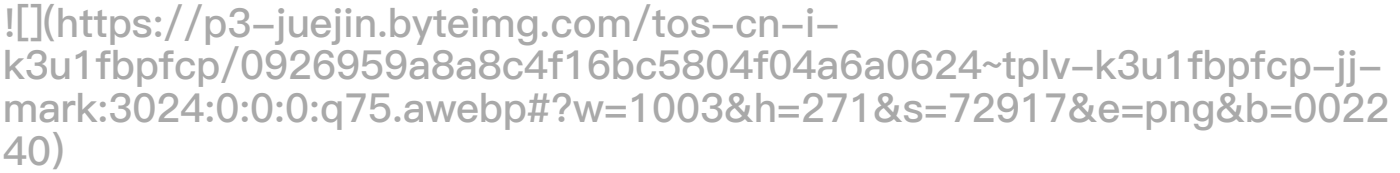
1、添加 SpringBoot Redis 依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

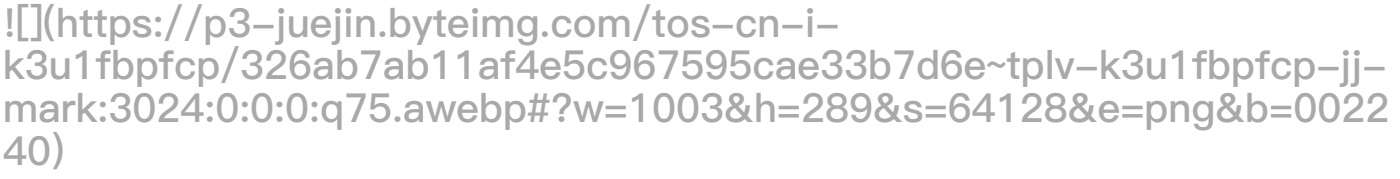
2、yaml 文件配置



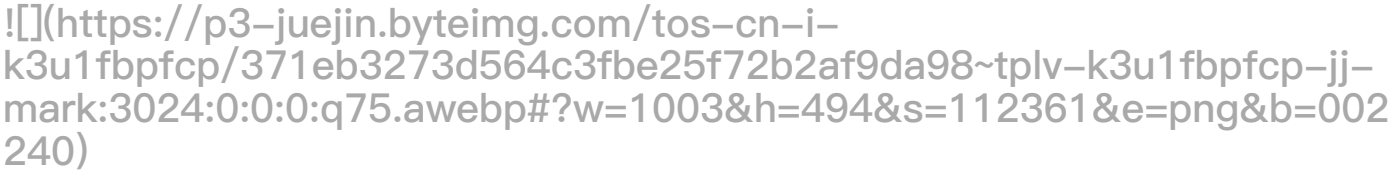
3、RedisTemplate 配置



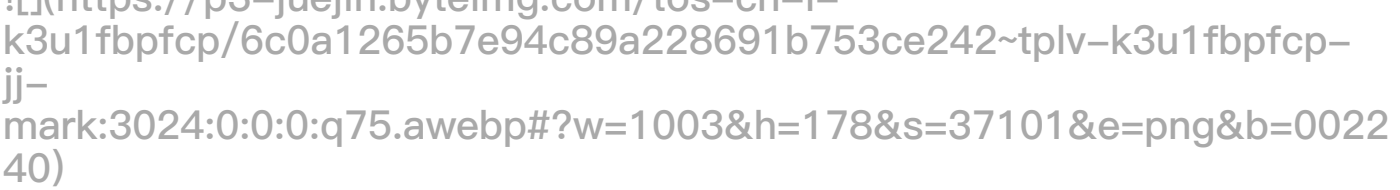
4、定义stream监听器



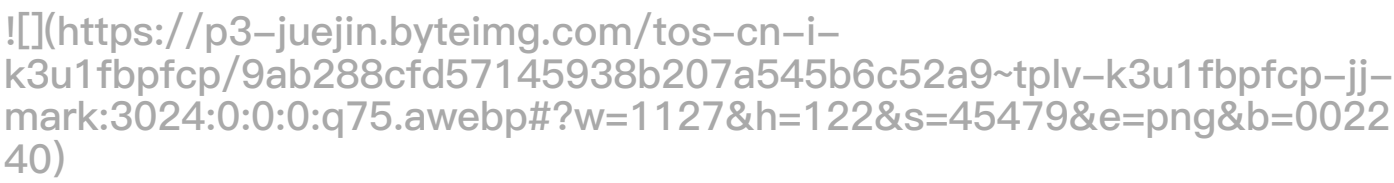
5、定义streamcontainer 并启动



6、发送消息



执行完成之后，消费者就可以打印如下日志：



演示代码地址：

> [github.com/makemyownli...](http://cxyroad.com/”https://github.com/makemyownlife/courage-cache-demo”)

4 Redis stream 用做消息队列完美吗

笔者认为 Redis stream 用于消息队列最大的进步在于：**实现了发布订阅模型**。

发布订阅模型具有如下特点：

* **消费独立**

相比队列模型的匿名消费方式，发布订阅模型中消费方都会具备的身份，一般叫做订阅组（订阅关系），不同订阅组之间相互独立不会相互影响。
* **一对多通信**

基于独立身份的设计，同一个主题内的消息可以被多个订阅组处理，每个订阅组都可以拿到全量消息。因此发布订阅模型可以实现一对多通信。

细品 Redis stream 的设计，我们发现它和 Kafka 非常相似，比如说消费者组，消费进度偏移量等。

我们曾经诟病 Redis List 数据结构用做队列时，因为消费时没有 Ack 机制，应用异常挂掉导致消息偶发丢失的情况，Redis Stream 已经完美的解决了。

因为消费者内部有一个属性 **pending_ids** ， 记录了当前消费者读取但没

有回复 ACK 的消息 ID 列表 。当消费者重新上线，这些消息可以重新被消费。

但 Redis stream 用做消息队列完美吗 ？

这个真没有！ 。

1、Redis 本身定位是**内存数据库**，它的设计之初都是为缓存准备的，**并不具备消息堆积的能力**。而专业消息队列一个非常重要的功能是**数据中转枢纽**，Redis 的定位很难满足，所以使用起来要非常小心。

2、Redis 的高可用方案可能丢失消息（AOF 持久化 和 主从复制都是异步），而专业消息队列可以针对不同的场景选择不同的高可用策略。

所以，笔者认为 Redis 非常适合轻量级消息队列解决方案，轻量级意味着：数据量可控 + 业务模型简单 。

参考文章：

> [redis.io/docs/data-t...](http://cxyroad.com/”https://redis.io/docs/data-types/streams/”)
>
>
> [www.runoob.com/redis/redis...](http://cxyroad.com/”https://www.runoob.com/redis/redis-stream.html”)
>
>
> [pdai.tech/md/db/nosql...](http://cxyroad.com/”https://pdai.tech/md/db/nosql-redis/db-redis-data-type-stream.html”)

如果我的文章对你有所帮助，还请帮忙**点赞、在看、转发**一下，你的支持会激励我输出更高质量的文章，非常感谢！

原文链接: https://juejin.cn/post/7357301805569687563