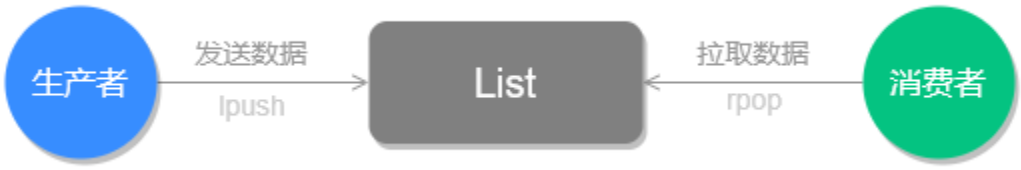
25 消息队列的其他实现方式

在 Redis 5.0 之前消息队列的实现方式有很多种，比较常见的除了我们上文介绍的发布订阅模式，还有两种：List 和 ZSet 的实现方式。

List 和 ZSet 的方式解决了发布订阅模式不能持久化的问题，但这两种方式也有自己的缺点，接下来我们一起来了解一下，先从 List 实现消息队列的方式说起。

List 版消息队列

List 方式是实现消息队列最简单和最直接的方式，它主要是通过 lpush 和 rpop 存入和读取实现消息队列的，如下图所示：



List 使用命令的方式实现消息队列：

```
127.0.0.1:6379> lpush mq "hello" #推送消息 hello
(integer) 1
127.0.0.1:6379> lpush mq "msg" #推送消息 msg
(integer) 2
127.0.0.1:6379> rpop mq #接收到消息 hello
"hello"
127.0.0.1:6379> rpop mq #接收到消息 msg
"mq"
```

其中，mq 就相当于频道名称 channel，而 lpush 用于生产消息， rpop 拉取消息。

代码实现

接下来我们用 Java 代码的方式来实现 List 形式的消息队列，源码如下：

```
import redis.clients.jedis.Jedis;

public class ListMQExample {
    public static void main(String[] args){
        // 消费者
        new Thread(() -> consumer()).start();
        // 生产者
        producer();
    }
    /**
     * 生产者
     */
    public static void producer() {
        Jedis jedis = new Jedis("127.0.0.1", 6379);
        // 推送消息
        jedis.lpush("mq", "Hello, List.");
    }
    /**
     * 消费者
     */
    public static void consumer() {
        Jedis jedis = new Jedis("127.0.0.1", 6379);
        // 消费消息
        while (true) {
            // 获取消息
            String msg = jedis.rpop("mq");
            if (msg != null) {
                // 接收到了消息
                System.out.println("接收到消息：" + msg);
            }
        }
    }
}
```

以上程序的运行结果是：

```
接收到消息：Hello, List.
```

我们使用无限循环来获取队列中的数据，这样就可以实时地获取相关信息了，但这样会带来另一个新的问题，当队列中如果没有数据的情况下，无限循环会一直消耗系统的资源，这时候我们可以使用 brpop 替代 rpop 来完美解决这个问题。

b 是 blocking 的缩写，表示阻塞读，也就是当队列没有数据时，它会进入休眠状态，当有数据进入队列之后，它才会“苏醒”过来执行读取任务，这样就可以解决 while 循环一直执行消耗系统资源的问题了，改良版代码如下：

```
import redis.clients.jedis.Jedis;
```

```
public class ListMQExample {
    public static void main(String[] args) throws InterruptedException {
        // 消费者 改良版
        new Thread(() -> bConsumer()).start();
        // 生产者
        producer();
    }
    /**
     * 生产者
     */
    public static void producer() throws InterruptedException {
        Jedis jedis = new Jedis("127.0.0.1", 6379);
        // 推送消息
        jedis.lpush("mq", "Hello, List.");
        Thread.sleep(1000);
        jedis.lpush("mq", "message 2.");
        Thread.sleep(2000);
        jedis.lpush("mq", "message 3.");
    }
    /**
     * 消费者（阻塞版）
     */
    public static void bConsumer() {
        Jedis jedis = new Jedis("127.0.0.1", 6379);
        while (true) {
            // 阻塞读
            for (String item : jedis.brpop(0,"mq")) {
                // 读取到相关数据，进行业务处理
                System.out.println(item);
            }
        }
    }
}
```

其中，brpop() 方法的第一个参数是设置超时时间的，设置 0 表示一直阻塞。

优缺点分析

List 优点：

- 消息可以被持久化，借助 Redis 本身的持久化（AOF、RDB 或者是混合持久化），可以有效的保存数据；
- 消费者可以积压消息，不会因为客户端的消息过多而被强行断开。

List 缺点：

- 消息不能被重复消费，一个消息消费完就会被删除；

- 没有主题订阅的功能。

ZSet 版消息队列

ZSet 版消息队列相比于之前的两种方式，List 和发布订阅方式在实现上要复杂一些，但 ZSet 因为多了一个 score（分值）属性，从而使它具备更多的功能，例如我们可以用它来存储时间戳，以此来实现延迟消息队列等。

它的实现思路和 List 相同也是利用 zadd 和 zrangebyscore 来实现存入和读取，这里就不重复叙述了，读者可以根据 List 的实现方式来实践一下，看能不能实现相应的功能，如果写不出来也没关系，本课程的后面章节，介绍延迟队列的时候会用 ZSet 来实现。

优缺点分析

ZSet 优点：

- 支持消息持久化；
- 相比于 List 查询更方便，ZSet 可以利用 score 属性很方便的完成检索，而 List 则需要遍历整个元素才能检索到某个值。

ZSet 缺点：

- ZSet 不能存储相同元素的值，也就是如果有消息是重复的，那么只能插入一条信息在有序集合中；
- ZSet 是根据 score 值排序的，不能像 List 一样，按照插入顺序来排序；
- ZSet 没有向 List 的 brpop 那样的阻塞弹出的功能。

小结

本文介绍了消息队列的另外两种实现方式 List 和 ZSet，它们都是利用自身方法，先把数据放到队列里，在使用无限循环读取队列中的消息，以实现消息队列的功能，相比发布订阅模式本文介绍的这两种方式的优势是支持持久化，当然它们各自都存在一些问题，所以期待下一课时 Stream 的出现能够解决这些问题。

上一页 下一页