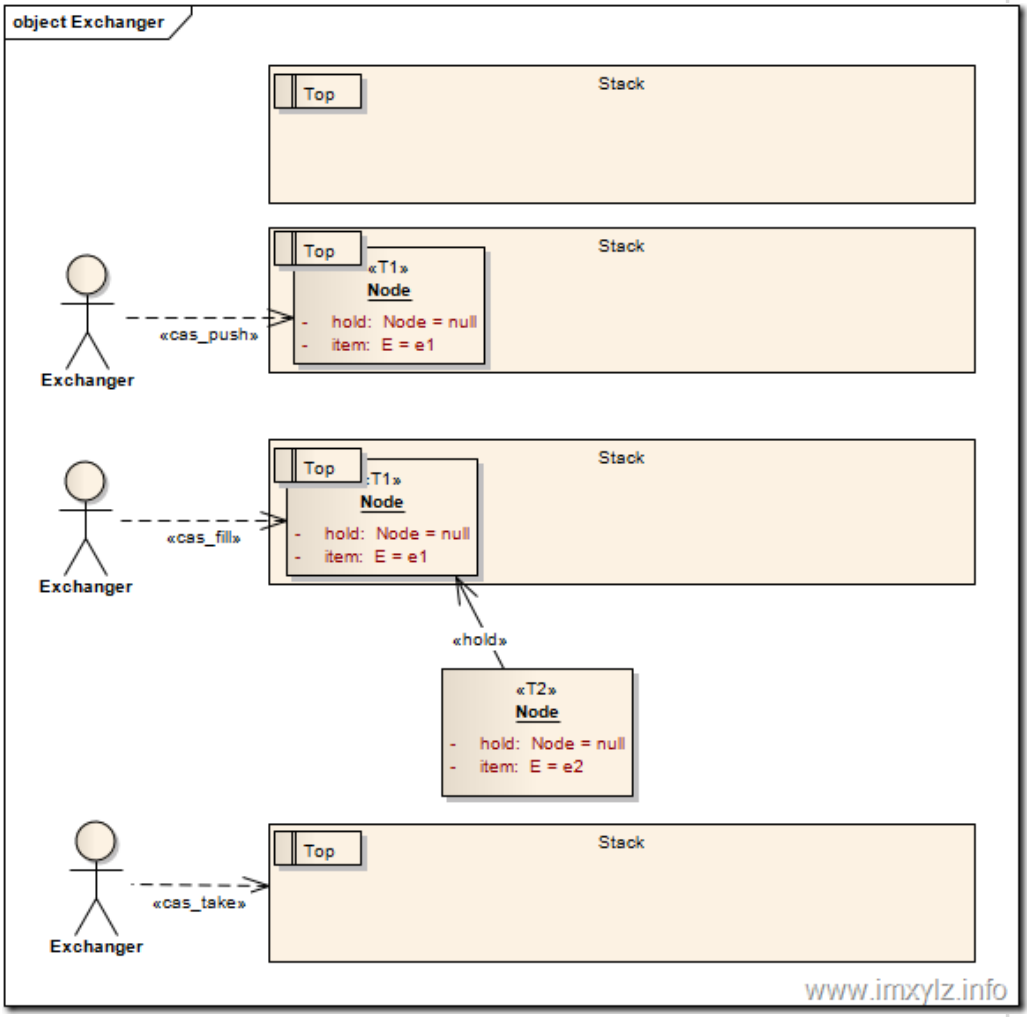
深入浅出 Java Concurrency (26): 并发容器 part 11 Exchanger

可以在对中对元素进行配对和交换的线程的同步点。每个线程将条目上的某个方法呈现给 exchange 方法，与伙伴线程进行匹配，并且在返回时接收其伙伴的对象。Exchanger 可能被视为 SynchronousQueue 的双向形式。

换句话说Exchanger提供的是一个交换服务，允许原子性的交换两个（多个）对象，但同时只有一对才会成功。先看一个简单的实例模型。



在上面的模型中，我们假定一个空的栈（Stack），栈顶（Top）当然是没有元素的。同时我们假定一个数据结构Node，包含一个要交换的元素E和一个要填充的"洞"Node。这时线程T1携带节点node1进入栈（cas_push），当然这是CAS操作，这样栈顶就不为空了。线程T2携带节点node2进入栈，发现栈里面已经有元素了node1，同时发现node1的hold（Node）为空，于是将自己（node2）填充到node1的hold中（cas_fill）。然后将元素node1从栈中弹出（cas_take）。这样线程T1就得到了node1.hold.item也就是node2的元素e2，线程T2就得到了node1.item也就是e1，从而达到了交换的目的。

算法描述就是下图展示的内容。

2011年1月 (5)
2010年12月 (4)
2010年11月 (4)
2010年8月 (3)
2010年7月 (24)
2010年6月 (2)
2010年1月 (5)
2009年12月 (12)
2009年11月 (2)
2009年9月 (1)
2009年7月 (1)

文章分类(12)
Eclipse (rss)
Java (rss)
Python(12) (rss)

文章档案(12)
2010年6月 (6)
2010年5月 (3)
2010年4月 (3)

友情链接
imxylz (rss)
jafka
a fast distributed publish-subscribe messaging system

搜索
搜索

积分与排名
积分 - 1918335
排名 - 4

最新评论 XML
1. re: 深入浅出 Java Concurrency (9): 锁机制 part 4
评论内容较长,点击标题查看
--guanzhisong
2. 提供参考链接
评论内容较长,点击标题查看
--33
3. re: 处理Zookeeper的session过期问题
评论内容较长,点击标题查看
--codewarrior
4. re: 当Ajax遭遇GBK编码 (完全解决方案)
前面说的都同意 就是解决方法说的太抽象 没看懂 给个具体的方法
--穆
5. re: 深入浅出 Java Concurrency (25): 并发容器 part 10 双向并发阻塞队列 BlockingDeque[未登录]
Mark,今天看到这里啦，满满的成就感。。。。。嘿嘿
--邓
6. order viagra
Hello!
--order_viagra
7. order cialis
Hello!
--order_cialis
8. cialis
Hello!
--cialis
9. dosage of viagra
Hello!
--of
10. viagra

```
00  Object exchange(Object x, boolean timed,
01    long patience) throws TimeoutException {
02    boolean success = false;
03    long start = System.nanotime();
04    Node mine = new Node(x);
05    for (;;) {
06      Node top = stack.getTop();
07      if (top == null) {
08        if (stack.casTop(null, mine)) {
09          while (null == mine.hole) {
10            if (timedOut(start, timed, patience)) {
11              if (mine.casHole(null, FAIL))
12                throw new TimeoutException();
13              break;
14            }
15            /* else spin */
16          }
17          return mine.hole.item;
18        }
19      } else {
20        success = top.casHole(null, mine);
21        stack.casTop(top, null);
22        if (success)
23          return top.item;
24      }
25    }
26  }
```


JDK 5就是采用类似的思想实现的Exchanger。JDK 6以后为了支持多线程多对象同时Exchanger了就进行了改造（为了支持更好的并发），采用ConcurrentHashMap的思想，将Stack分割成很多的片段（或者说插槽Slot），线程Id（Thread.getId()）hash相同的落在同一个Slot上，这样在默认32个Slot上就有很好的吞吐量。当然会根据机器CPU内核的数量有一定的优化，有兴趣的可以去了解下Exchanger的源码。

至于Exchanger的使用，在JDK文档上有个例子，讲述的是两个线程交换数据缓冲区的例子（实际上仍然可以认为是生产者/消费者模型）。

```
class FillAndEmpty {
  Exchanger<DataBuffer> exchanger = new Exchanger<DataBuffer>();
  DataBuffer initialEmptyBuffer = … a made-up type
  DataBuffer initialFullBuffer = …

  class FillingLoop implements Runnable {
    public void run() {
      DataBuffer currentBuffer = initialEmptyBuffer;
      try {
        while (currentBuffer != null) {
          addToBuffer(currentBuffer);
          if (currentBuffer.isFull())
            currentBuffer = exchanger.exchange(currentBuffer);
        }
      } catch (InterruptedException ex) { … handle … }
    }
  }

  class EmptyingLoop implements Runnable {
    public void run() {
      DataBuffer currentBuffer = initialFullBuffer;
      try {
        while (currentBuffer != null) {
          takeFromBuffer(currentBuffer);
          if (currentBuffer.isEmpty())
            currentBuffer = exchanger.exchange(currentBuffer);
        }
      } catch (InterruptedException ex) { … handle …}
    }
  }

  void start() {
    new Thread(new FillingLoop()).start();
    new Thread(new EmptyingLoop()).start();
  }
}
```