# DelayQueue的使用

## 应用场景

DelayQueue可以实现一个延时队列，应用场景包括

1. 某个条件触发后延时多久要保持一个状态，直到超时之后才会释放
2. 某个条件触发后需要延迟多久触发另外一个条件

这是一个比较有趣的特性，如果运用的好可以实现很多巧妙的效果。

## 使用方法

需要先定义一个DelayItem对象，此对象必须实现Delayed接口

public class DelayItem implements Delayed {

private long delayTime;

/\*

\* (non-Javadoc)

\*

\* @see java.util.concurrent.Delayed#getDelay(java.util.concurrent.TimeUnit)

\*/

@Override

public long getDelay(TimeUnit unit) {

return delayTime - System.currentTimeMillis();

}

}

将DelayItem放入DelayQueue中只有在DelayItem超时之后才能通过take、poll方法获取到

public class SlotEventParser {

private DelayQueue<SlotEventDelayItem> slotProvFaildDelayQueue;

。。。。。。

// 初始化线程队列

slotOfflineDelayQueue= new DelayQueue<SlotEventDelayItem>();

//主动移除delayQueue中的超时对象

new Thread () {

public void run() {

while (true) {

try {

SlotEventDelayItem delayItem = slotOfflineDelayQueue.take();

logger.debug("SlotEventParser slotOfflineDelayQueue free delayItem[" +delayItem+ "]");

} catch (InterruptedException e) {

logger.debug("",e);

}

try {

Thread.sleep(10000);

} catch (InterruptedException e) {

}

}

}

}.start();

}

SlotEventDelayItem delayItem = new SlotEventDelayItem(slotOfflineDelayed\_time \* 1000L, event);

if (!SlotEventDelayItem .contains(delayItem)) {

//如果还未超时，就不能向slotQueue中添加内容，用DelayQueue保存了状态

slotOfflineDelayQueue.put(delayItem);

slotQueue.add(event);

}

}