在这之后，Source 的 doStart 方法被调用。这种方法可以用于建立任何网络客户端。在这种情况下，获取股票报价的服务就启动了 (因为这只是一个例子，在这种情况下，服务只是返回随机值)。

然后 Pollable Source 运行器调用 process 方法（在 BasicSourceSemantics 实现过），这也表示调用了 doProcess 方法。doProcess 方法利用外部服务获取的数据来生成事件。当 Source 成功将传入的引用转换为 Flume 事件，并且通过 Channel 处理器的 processEventBatch 方法将事件写入 Channel 时，它会返回 Status.READY；否则，它将返回 Status.BACKOFF。在这种情况下，如果在比刷新间隔更少的时间连续调用 doProcess，Source 只会返回 Status.BACKOFF，这表明 Source 运行器再次调用 process 方法时应该等待一段时间。

最终，在 Agent 停止时，doStop 方法就被调用。这个方法可以做任何清理，如关闭网络连接。一旦停止代理，该实例可以被垃圾回收。

下面是 Pollable Source 一个简单的例子。如果想看更真实的例子，请查看 Flume[jms_src_code] 的 JMS Source。

创建 Event-driven Source

Event-driven sources 实现了 EventDrivenSource 接口，对于 Flume 框架这仅仅是选择 SourceRunner 实现来运行 Source 的一个标记接口。当 Flume 框架调用 start 方法，Event-driven Source 通常会运行它们自己的线程。这类 Source 控制了它们写数据到 Channel 的速率。

74

例如，Flume 自带的 HTTP Source 就是一个 event-driven Source，它运行一个 web 监听特定端口的服务器。它基于发送给 HTTP 请求的事件来生成 Flume 的事件，并将这些事件写入与之关联的 Channel。Event-driven Source 通常运行它们自己的线程或线程池，用来处理事件生成和事件写入到 Channel。因为这些 Source 要对一些外部刺激做反应，Flume 框架创建一个新的 EventDrivenSourceRunner，通过在新线程调用 start 方法来启动这些 Source，并允许它们自己来管理。当 Agent 停止或重新配置时，调用 stop 方法来停止 Source。

Event-driven Source 响应外部事件来产生数据。大多数从外部实体来接收数据的 Source 都属于这一类。Event-driven Source 运行自己的线程用于接收数据并生成事件。大部分 Flume 自带的 Source，例如 Avro Source、HTTP Source、Exec Source 等，都是 Event-driven Source。