每个 Source“生成”事件，并为了 Source 转发事件到 Channel 处理器中。Source 每次生成一个事件，调用 Channel 处理器的 processEvent 方法将事件写入 Channel 处理器，或者使用 Channel 处理器的 sprocessEventBatch 方法来发送事件。使用 processEventBatch 方法来处理一批事件总是更好的。每次 processEventBatch 调用启动一个 Channel 事务，每次事务写入整个批次再提交。另一方面，processEvent 只为一个事务创建事务，这可能会导致严重的开销，影响 Channel 的性能。这就是为什么建议使 Source 使用 processEventBatch 方法，除非已知每个事务都很大（近似数百千字节到几兆字节）。要访问 Source 的 Channel 处理器，Source 可以调用 AbstractSource 类中定义的 getChannelProcessor 方法。

如果来自外部 Source 的数据需要认证后被发送，重要的是这只在 processEventBatch 方法返回后被发送。如果对 channel 的提交失败，我们必须通知原始数据源，需要再次发送数据。在 processEventBatch 方法返回后发送 ACK 可以避免这个问题。这个方法抛出的 ChannelExceptions 可以被捕获，且失败可以报告给数据源，所以数据就可以重新发送。

通过 Flume 框架每个 Source 的 Channel 处理器被创建和启用，所以 Source 不需要处理 Channel 处理器的创建或配置工作。在这一部分，我们将了解如何写自定义 Source。

这部分所有被描述为依赖的类都是 flume-ng-core 工件或它的依赖的一部分。例 3-6 描述了如何在你插件的 *pom.xml* 文件中包含这个工件。就像第 8 章“部署自定义代码”一节中展示的，自定义 Source 可以像其他插件一样被部署到 Flume。

71 Event-Driven Source 和 Pollable Source

每个 Source 在称为 SourceRunner 的自身线程上运行。Source 运行器运行单独的线程来操作 Source。Flume 有两种类型的 Source：*Event-driven Source* 和 *Pollable Source*。在 Source 类型的基础上，Flume 框架会创建 EventDrivenSourceRunner 或 PollableSourceRunner 来运行 Source。

开发 Pollable Source

Pollable Source 不运行它们自己的线程；它们反而受 Flume 框架的控制，即 Flume 框架会循环调用 Source 的 process 方法。这些 Source 继承了 AbstractPollableSource 类并实现了 process 方法。Pollable Source 可以通过 Flume 配置系统接收用户的配置，这种功能除了要继承 AbstractPollableSource 类还要实现 Configurable 接口。

Pollable Source 运行一个循环来生成数据或轮询外部系统来接收数据，而不是运行一个服务器。一旦配置提供者实例化并且配置了 Pollable Source，Flume 框架会创建一个