```
      e.setCharset(charset);
      newEvents.add(EventBuilder.withBody(e.getBody(),
        e.getHeaders()));
    }
    return newEvents;
  }

  @Override
  public void configure(Context context) {
    try {
    charset = context.getString("charset", "UTF-8");
    } catch (Exception ex) {
      LOGGER.warn("Charset not found. Using UTF-8 instead", ex);
      charset = "UTF-8";
    }

  }
}
```

例 3-13 中展示的转换器期望是 JMS TextMessage 的消息。该消息被期望是 JSON 格式转换为一系列 Flume 事件。一旦 JSON 事件被创建，一系列“simple”的 Flume 事件也随之创建，以避免每次读取的时候将字符串转换为字节数组产生的额外开销。要为 JMS Source 配置转换器，可以使用例 3-11。

要部署一个自定义的转换器，首先要确保 *plugins.d* 目录包含这个类的 JAR 文件，然后要使用 converter.type 参数指定类的 FQCN。当写你的反序列化器时，要在你应用程序的 *pom.xml* 文件中包含 flume-jms-source 工件。

```
        <dependency>
          <groupId>org.apache.flume.flume-ng-sinks</groupId>
          <artifactId>flume-hdfs-sink</artifactId>
          <version>1.5.0</version>
    </dependency>
```

自定义的转换器可以像第 8 章“部署自定义代码”一节中叙述的一样安装到 *plugins.d* 目录。

编写自定义 Source*

Source 是数据进入到 Flume Agent 的要点。用户很可能会需要使用定制的或专有的通信格式用于写数据到 Flume。通过 Flume SDK 推送数据通常是更高效、更容易的。如果要与其他系统集成，用户可以编写自己的 Flume Source，并使用 Flume 的 *plugins.d* 机制部署它们。

70