正确 Source 的正确 Channel 处理器可以传递事件到正确的拦截器（虽然从配置来看，拦截器看起来像是 Source 的子组件——Source 实际上并不需要创建或配置拦截器）。类似地，每个 Sink 的 Channel 也由配置系统来设置。配置系统增加 Sink 到正确的 Sink 组，还为 Sink 组配置 Sink 处理器。

需要从 Flume 配置系统获取配置的任何组件，都必须实现 Configurable 接口，如例 2-1 所示。

例2-1 Configurable接口

```
package org.apache.flume.conf;
public interface Configurable {
  public void configure(Context context);
}
```

组件可具有子组件，子组建也可以是可配置的。每个组件必须配置其子组件或传递子组件的配置到子组件，然后子组件必须进行自配置。虽然每个子组件可以用组件指定实现的任何方式进行配置，但是实现 Configurable 接口是一种很好的做法。使用 Context 类的 getSubProperties 方法，可以将指定给子组件的子属性传递给子组件。

配置子组件的一个例子是 HDFS 序列化器，在第 5 章的“使用序列化器控制数据格式 *”一节中有详细描述。序列化器通过使用 Configurable 接口和 getSub Properties 方法，由 HDFS Sink 配置。序列化器可以使用后缀 serializer. 对 hdfsSink 进行配置。在配置文件中，序列化器获取 serializer. 后面的子串作为键。在下面的例子中，序列化器将获取带一个键值对的 Context 实例。键是 bufferSize 且值是 4096：

```
agent.sinks.hdfsSink.serializer.bufferSize = 4096
```

带可配置子组件的所有 Flume 组件都遵循这种模式，每个子组件只获取移除所有前缀的自己的参数。这对于子组件的子组件也适用，前提是有子组件存在。

例 2-2 展示了具有多个组件的 Flume Agent 的一个例子，其中一些组件拥有子组件。在该 Agent 中，有一个 Source、两个 Channel 和两个 Sink。Source 是一个 HTTP Source，命名为 httpSrc。该 Source 写入两个内存 Channel，memory1 和 memory2——这是由配置系统在 Channel 处理器中设置，Source 实现不需要设置 Channel。多个参数——bind、port、ssl、keystore、eystore-password、handler 和 handler.insertTimestamp——它们的值在传递给配置方法的 Context 实例中是可用的。Source 的实现决定了传递给它的任何配置参数都负责什么事情。

对于此配置文件，配置系统还创建了一个拦截器，用于 HTTP Source 接收到的所有事件的转发。在该实例中，HTTP Source 不需要理会拦截器创建或配置的任何特殊处理。拦