Bolt消息处理者

认识了消息源Spout和消息的数据存储元组Tuple，接下来了解消息的处理这Bolt。

Bolt时接收Spout发出元组Tuple后处理数据的组件，所有的消息处理逻辑被封装在Bolt中，Bolt负责处理输入的数据流并产生输出的新数据流。

1. Bolt介绍

消息处理者Bolt在Storm中是一个被动的角色。Bolt把元组作为输入，然后产生新的元组作为输出。

1. Bolt的功能

Bolt可以执行过滤、函数操作、合并、写数据库等操作。Bolt还可以简单地传递消息流，复杂的消息流处理往往需要很多步骤，因此也就需要很多Bolt来处理。

Bolt可以发出超过一个的流。为此，使用OutputFieldsDeclarer类的declareStream()方法声明多个流，并使用OutputCollector类的emit()方法指定发射的流。

1. Bolt的生命周期

首先，客户的那机器创建Bolt，然后将其序列化为拓扑，并提交给集群中的主机。之后集群启动Worker进程，反序列化Bolt，调用prepare方法开始处理元组。

接下来，Bolt处理Tuple，Bolt处理一个输入Tuple，发射0个或者多个Tuple。

然后，调用ack通知Storm自己已经处理过这个Tuple了。Storm提供了一个IBasicBolt自动调用ack。

Bolt类接收由Spout或者其他上游Bolt类发来的Tuple，对其进行处理。Bolt的生命周期如图所示。



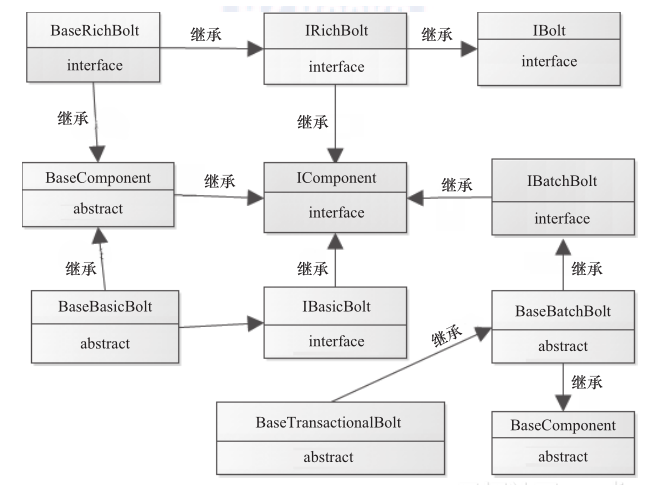
在创建Bolt对象时，通过构造方法和初始化成员变量，当Bolt被提交到集群时，这些成员变量也会被序列化，所以通过反序列化，可以获取到这些成员变量。

1. Bolt的组件

IComponent顾名思义，时所有组件的接口：IBasicBolt、IRichBolt、IBatchBolt都继承自IComponent；

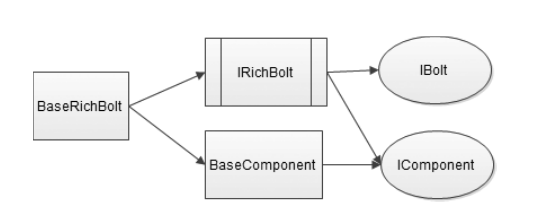
IBolt接口是IRichBolt要继承的接口；

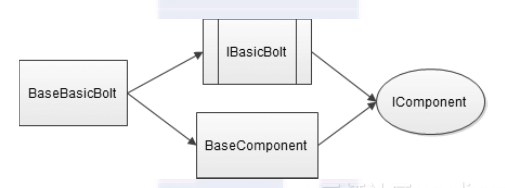
还有一些以Base开头的Bolt类，如BaseBasicBolt、BaseBatchBolt、BaseRichBolt、BaseTransactionBolt等，在这些类中需要注意的是所实现的方法都为空，或者返回值为null，其中，还有一个接口BaseComponent，是Storm提供的一个比较方便的抽象类，这个抽象类及其子类都或多或少实现了其接口定义的部分方法。从下图中，可以从整体上看到这些类的关系图，从而理清这些类之间的关系及结构。



1. Bolt的常用类

Bolt比较常用的类是BaseRichBolt、BaseBasicBolt等。这两个类继承的父类如下图所示，它们的共同之处是父类中都有BaseComponent和IComponent。不同之处是BaseRichBolt的父接口中有IBolt和IRichBolt，而BaseBasicBolt只有IBasicBolt。

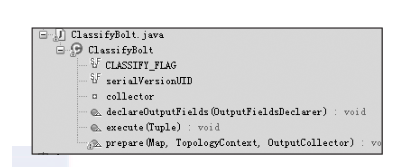




IBolt继承了java.io.Serializable，在Nimbus上提交Topology以后，创建出来的Bolt在序列化后被发送到具体执行的Worker上，Worker在执行该Bolt时，先调用prepare方法传入当前执行的上下文，然后调用execute方法，对Tuple进行处理，并用prepare方法传入的OutputCollector的ack方法（表示成功）或fail方法（表示失败）来反馈处理结果。而IBasicBolt接口在执行execute方法时，自动调用ack方法，其目的就是实现该接口的Bolt时，不用在代码中提供反馈结果，Storm内部会自动反馈成功。

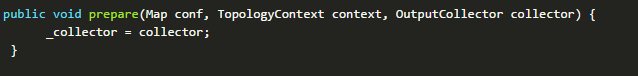
1. Bolt实例

下面的ClassifyBolt实现了BaseRichBolt接口，该类需要实现的主要方法如图所示。



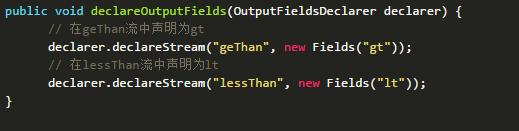
1. prepare方法

prepare方法和Spout中的open方法类似，为Bolt提供了OutputCollector，用来从Bolt中发送Tuple。在Bolt中载入新的线程进行异步处理。OutputCollector时线程安全的，并且随时都可以调用它。在Bolt中，Tuple的发送可以在prepare、execute、cleanup等方法中进行，但一般都是在execute中进行。



1. declareOutputFields方法

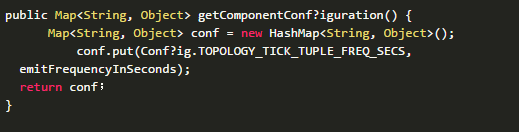
用于声明当前Bolt发送的Tuple中包含的字段，和Spout中的类似。当前Bolt类发送的Tuple包含了两个字段：gt和lt。示例代码如下：



Bolt可以发射多条消息流，使用OutputFieldsDeclarer.declareStream方法来定义流，之后使用OutputCollector.emit来选择要发射的流。

1. getComponentConfiguration方法

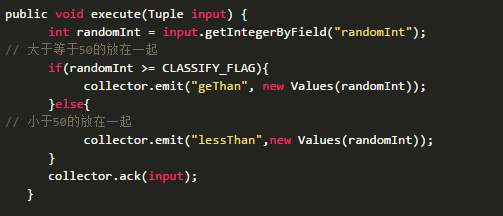
和Spout类一样，在Bolt中也可以有getComponentConfiguration方法。示例代码如下：



此例定义了从系统组件“\_system”的”\_tick”流中发送Tuple到当前Bolt的频率，当系统需要每隔一段时间执行特定的处理时，就可以利用该系统组件的特性来完成。

1. execute方法

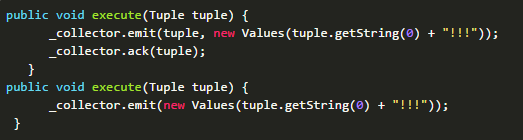
Bolt的主要方法时execute，它以一个Tuple作为输入，Bolt使用OutputCollector来发射Tuple，Bolt必须为它处理的每一个Tuple调用OutputCollector的ack方法，以通知Storm该Tuple被处理完成了，从而通知该Tuple的发射者Spout。



execute是Bolt中最关键的一个方法，对Tuple的处理都可以放到此方法中进行。具体的发送也是通过emit方法来完成的。此时，emit方法有两种情况，一种是方法中只有一个参数，另一种是方法中有两个参数。

1. emit有一个参数：该参数是发送到下游Bolt的Tuple，此时，由上游发来的旧的Tuple在此隔断，新的Tuple和旧的Tuple不再属于同一棵Tuple树。新的Tuple另起一棵新的Tuple树。
2. emit有两个参数：第一个参数是旧的Tuple的输入流，第二个参数是发往下游Bolt的新的Tuple流。此时，新的Tuple和旧的Tuple仍然属于同一棵Tuple树，即如果下游的Bolt处理Tuple失败，则向上传递到当前Bolt，当前Bolt根据旧的Tuple继续往上游传递，申请重发失败的Tuple，保证Tuple处理的可靠性。

这两种情况都要根据用户的场景来确定。示例代码如下：



此外还有ack、fail、cleanup等方法，其中cleanup方法和Spout中的close方法类似，都是在当前组件关闭时调用，但是针对实时计算来说，除非一些特殊的场景要求以外，这两个方法一般都很少用到。

