### Evctors

编程过程当中，可以通过调用DataStream API的evictor()方法“可选方法”传入相应的Evictor对进入WindowFunction前后的数据进行剔除处理，默认的Evictors都是在WindowFunction计算之前对数据进行剔除处理的。

Flink本身实现了三种Evictor，其中有CountEvictor，DeltaEvictor和TimeEvictor。

### 2.CountEvictor

CountEvictor

定义：在窗口中保持固定数量的数据，将超过指定大小的数据在窗口计算之前剔除。

demo:仅保留窗口中5条数据

.evictor(CountEvictor.of(5)).

开发：

**package** com.sqq2;  
  
**import** org.apache.flink.api.common.functions.FlatMapFunction;  
**import** org.apache.flink.api.java.tuple.Tuple2;  
**import** org.apache.flink.streaming.api.TimeCharacteristic;  
**import** org.apache.flink.streaming.api.datastream.DataStream;  
**import** org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;  
**import** org.apache.flink.streaming.api.windowing.evictors.CountEvictor;  
**import** org.apache.flink.util.Collector;  
  
**public class** wr8 {  
  
 **public static void** main(String[] args) **throws** Exception {  
 StreamExecutionEnvironment env = StreamExecutionEnvironment.*getExecutionEnvironment*();  
 env.setStreamTimeCharacteristic(TimeCharacteristic.***ProcessingTime***);  
  
 DataStream<Tuple2<String, Integer>> dataStream = env  
 .socketTextStream(**"192.168.8.201"**,9999)  
 .flatMap(**new** com.sq.wc1.Sp())  
 .keyBy(0)  
 .countWindow(10)  
 .evictor(CountEvictor.*of*(5))  
 .sum(1);  
  
 dataStream.print();  
  
 env.execute(**"Window WordCount"**);  
 }  
  
 **public static class** Sp **implements** FlatMapFunction<String, Tuple2<String, Integer>> {  
 @Override  
 **public void** flatMap(String sentence, Collector<Tuple2<String, Integer>> out) **throws** Exception {  
 **for** (String word: sentence.split(**" "**)) {  
 out.collect(**new** Tuple2<String, Integer>(word, 1));  
 }  
 }  
 }  
  
}

**结果：**

**Nc 输入**

[hadoop@h201 ~]$ nc -l 9999

haha haha

hehe hehe

haha

haha haha haha

hehe hehe

haha haha

hehe hehe

haha haha haha

查看结果：

只保留了5条

scala> (haha,5)

### 3.TimeEvictor

概念：通过指定时间间隔，将当前窗口中最新元素的时间减去Interval，然后将小于该结果的数据全部剔除，其本质是将具有最新时间的数据选择出来，删除过时的数据。

demo：仅保留最新2s内数据

.evictor(TimeEvictor.of(Time.seconds(2)))

开发：

**package** com.sqq2;  
  
**import** org.apache.flink.api.common.functions.FlatMapFunction;  
**import** org.apache.flink.api.java.tuple.Tuple2;  
**import** org.apache.flink.streaming.api.TimeCharacteristic;  
**import** org.apache.flink.streaming.api.datastream.DataStream;  
**import** org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;  
**import** org.apache.flink.streaming.api.windowing.evictors.TimeEvictor;  
**import** org.apache.flink.streaming.api.windowing.time.Time;  
**import** org.apache.flink.util.Collector;  
  
**public class** wr81 {  
  
 **public static void** main(String[] args) **throws** Exception {  
 StreamExecutionEnvironment env = StreamExecutionEnvironment.*getExecutionEnvironment*();  
 env.setStreamTimeCharacteristic(TimeCharacteristic.***ProcessingTime***);  
  
 DataStream<Tuple2<String, Integer>> dataStream = env  
 .socketTextStream(**"192.168.8.201"**,9999)  
 .flatMap(**new** com.sq.wc1.Sp())  
 .keyBy(0)  
 .timeWindow(Time.*seconds*(10))  
 .evictor(TimeEvictor.*of*(Time.*seconds*(2)))  
 .sum(1);  
  
 dataStream.print();  
  
 env.execute(**"Window WordCount"**);  
 }  
  
 **public static class** Sp **implements** FlatMapFunction<String, Tuple2<String, Integer>> {  
 @Override  
 **public void** flatMap(String sentence, Collector<Tuple2<String, Integer>> out) **throws** Exception {  
 **for** (String word: sentence.split(**" "**)) {  
 out.collect(**new** Tuple2<String, Integer>(word, 1));  
 }  
 }  
 }  
  
}

### DeltaEvictor

阈值剔除器。计算Window中最后一个元素与其余每个元素之间的增量，丢弃增量大于或等于阈值的元素。

a.先找到当前窗口的最后一条元素。

b.遍历窗口中的每一条元素。每条元素(A)和最后一条元素(L)，依据用户提供的DeltaFunction计算出一个Delta。计算出的Delta大于等于设定的阈值，则剔除该元素(A)。