### 1.dataset 与datastream区别

DataSet同DataStream从其接口封装、真实计算Operator有很大的差别，Dataset的实现在flink-javamodule中，而DataStream的实现在flink-streaming-java中；

DataSet： 批式处理，其接口封装类似于Spark的Dataset，支持丰富的函数操作，比如map/fliter/join/cogroup等；

数据源创建初始数据集，例如来自文件或Java集合等静态数据；

所有的操作为Operator的子类，实现具体逻辑，比如Join逻辑是在JoinOperator中实现；

DataStram: 流式处理，其结构封装实现输入流的处理，其也实现了丰富的函数支持

例子1：

**import** org.apache.flink.api.java.ExecutionEnvironment;  
**import** org.apache.flink.api.java.operators.DataSource;  
**import** org.apache.flink.api.java.tuple.Tuple2;  
  
**public class** ds1 {  
 **public static void** main(String[] args) **throws** Exception {  
 ExecutionEnvironment env = ExecutionEnvironment.*getExecutionEnvironment*();  
 DataSource<Long> source1 = env.generateSequence(1, 100);  
 source1.filter(**new** FilterFunction<Long>() {  
 @Override  
 **public boolean** filter(Long aLong) **throws** Exception {  
 **if**(aLong >90) {  
 **return true**;  
 }  
 **return false**;  
 }  
 }).map(**new** MapFunction<Long, Tuple2<String,Long>>() {  
 @Override  
 **public** Tuple2<String, Long> map(Long aLong) **throws** Exception {  
 **return new** Tuple2<>(**"haha"**,aLong);  
 }  
 }).print();  
 }  
}

2.

在某些算法中，可能需要为数据集元素分配惟一标识符

### 2.1zipWithIndex压缩密集索引

zipWithIndex为数据元分配连续标签，接收数据集作为输入并返回(unique id, initial value)2元组的新数据集。此过程需要两次传递，首先计算然后标记数据元，并且由于计数的同步而不能流水线化。该替代方案zipWithUniqueId以流水线方式工作，并且在唯一标签足够时是优选的。

**package** com.ds;  
**import** org.apache.flink.api.java.DataSet;  
**import** org.apache.flink.api.java.ExecutionEnvironment;  
**import** org.apache.flink.api.java.tuple.Tuple2;  
**import** org.apache.flink.api.java.utils.DataSetUtils;  
  
**public class** ds2 {  
 **public static void** main(String[] args)**throws** Exception{  
 ExecutionEnvironment env = ExecutionEnvironment.*getExecutionEnvironment*();  
 env.setParallelism(2);  
 DataSet<String> in = env.fromElements(**"A"**, **"B"**, **"C"**, **"D"**, **"E"**, **"F"**, **"G"**, **"H"**);  
 DataSet<Tuple2<Long, String>> result = DataSetUtils.*zipWithIndex*(in);  
 result.writeAsCsv(**"c:\\a.csv"**, **"\n"**, **","**);  
 env.execute();  
 }  
}

### 2.2zipWithUniqueId唯一标识符进行压缩

使用唯一标识符进行压缩

在许多情况下，可能不需要分配连续标签。 zipWithUniqueId以流水线方式工作，加快标签分配过程。此方法接收数据集作为输入，并返回(unique id, initial value)2元组的新数据集

**package** com.ds;  
  
**import** org.apache.flink.api.java.DataSet;  
**import** org.apache.flink.api.java.ExecutionEnvironment;  
**import** org.apache.flink.api.java.tuple.Tuple2;  
**import** org.apache.flink.api.java.utils.DataSetUtils;  
  
**public class** ds3 {  
 **public static void** main(String[] args)**throws** Exception{  
 ExecutionEnvironment env = ExecutionEnvironment.*getExecutionEnvironment*();  
 env.setParallelism(2);  
 DataSet<String> in = env.fromElements(**"A"**, **"B"**, **"C"**, **"D"**, **"E"**, **"F"**, **"G"**, **"H"**);  
 DataSet<Tuple2<Long, String>> result = DataSetUtils.*zipWithUniqueId*(in);  
  
 result.writeAsCsv(**"c://b.csv"**, **"\n"**, **","**);  
 env.execute();  
 }  
}