第一种方法的优点：1，抽取效率较快；2，封装性较好；缺点：1，开发的时候需要进行很多操作，很麻烦

第二种方法的优点：1，可移植性强，不用改动太多代码；2，开发时需要的步骤少，很快捷，修改代码方便；缺点：1，封装性差，抽取效率不高。

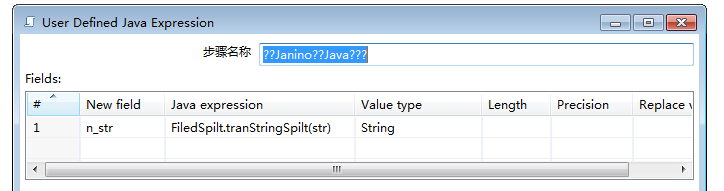
一、使用自定义的Java代码操作字符串的两种方式。

1. 将自己写好的Java代码打成jar包

(现将源文件变异成字节码文件，在将字节码文件打成jar包)

命令如下：jar cvf xxx.jar xxx.class

1. 将jar包放在kettle的data-integration/lib 目录下
2. 在转换中使用"Janino计算Java表达式"这个控件



方法名称

类名

新字段名

1. 利用Java代码控件实现

//导入所需要的jar包

import java.util.\*;

import java.lang.String;

private int inputStrIndex;

public boolean processRow(StepMetaInterface smi, StepDataInterface sdi) throws KettleException

{

  Object[] r=getRow();//获取传入的字段值

  if (r==null)

  {

    setOutputDone();

    return false;

  }

  //第一次访问该JAVA类的时候进行是否有传入参数的判断

  if (first) {

     //获取传入参数的访问位置

     inputStrIndex = getInputRowMeta().indexOfValue(getParameter("INPUT\_STR"));

     if (inputStrIndex<0) {

         throw new KettleException("INPUT\_STR field not found in the input row, check parameter 'INPUT\_STR'!");

     }

     first=false;

  }

  //数组扩容，因为有传入的参数，现在又有了新传入的参数，意味着至少要传入一个新的参数。这里outputRowMeta就是传出去的元数据集

  Object[] outputRowData = RowDataUtil.resizeArray(r, data.outputRowMeta.size());

  //这里获取传入参数的数组大小。其实outputIndex就是传入的参数的最后一个字段所在的位置。下面使用了outputIndex++。数组扩容了。那么如果传出一个参数的话，新的传出值得数组下标就是outputIndex+1。两个就是+2

  int outputIndex = getInputRowMeta().size();

  //获取传入参数的数值

  String inputStr = getInputRowMeta().getString(r, inputStrIndex);

  //这里给传出的参数赋值。stringUtil是我自定义的方法。

  outputRowData[outputIndex++] = stringUtil(inputStr.toString());

  //这里就是将新的数组内容赋给传出的元数据集。也就是将所有传出的字段堪称一个叫做outputRowMeta的数组。

  putRow(data.outputRowMeta, outputRowData);

  return true;

}

//这里就是写了具体的处理方法。

private String stringUtil(String inputStr) {

}