Java 调度 kettle 时，JNDI 数据源动态配置。

作者：someone9891
原文链接：https://ld246.com/article/1563503085558
来源网站：链滴

重要更新：

这种方式根本就不能实现我的业务，都是上千个定时任务并发执行的，并发去修改文件，亏我想的出，研究半天白折腾了

引

前一段时间由于kettle数据源的问题，写了一篇博客，连接如下：

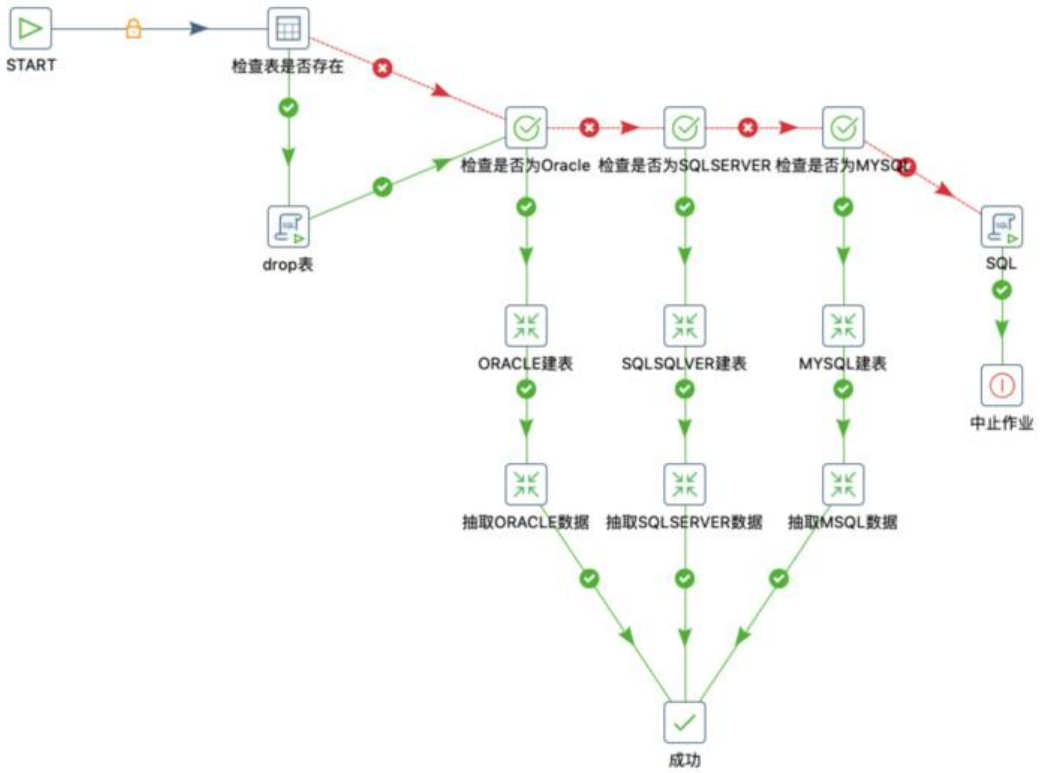
Java 调度 Kettle 时使用 jndi 连接数据库

上一篇博客的内容，解决了通过Java调用kettle时，覆盖默认的jndi 数据源配置文件地址，实现自己项目里进行配置数据源的需求。

kettle对于不同数据源的处理

在kettle中，在一个流程里是可以使用多种数据源的，比如 **表输入** 组件从 一个 mysql表中获取数据可以通过 **表输出** 组件输出到另一个 Oracle数据库中。

但是，有一个一直让我很纠结的问题，就是我搞一个通用的 流程，比如：



上面的流程比较简单，实现简单的从 X 库 到 A 库的数据抽取，A库是固定的，X 库是动态的，类型连接信息都是动态的，

这里，分了三个分支来处理三种不同的数据源，建表部分由于sql 语法不同，所以无论如何都是要三数据源单独处理的。

但是下面 这三个，其实是完全一样的，但是 由于连接数据库不一样，所以还是得做成三个：

下面看一下kettle数据源连接的方式：

这是kettle新建数据源的窗口，可以看出来，数据源的连接信息是填写的，并且是可以动态使用变量。

但是数据库驱动从左边选择，无法动态传值。

根据 http://blog.gitor.org/articles/2019/06/29/1561815612401 里面说明，可以使用 JNDI 数据来解决动态传人驱动名的问题，

但是JNDI 是读取的配置文件，就算配置多种数据源驱动，但是数据库连接信息如果存储在数据库之的，也是没办法动态传入的。

设想

在kettle执行前，先会进行初始化，这期间就会初始化JNDI数据源，那么是否可以在初始化之前 对 jnd 的 jdbc.properties 文件进行修改，然后kettle他还是自己去读这个文件呢?

实践

首先，由于初始化的时候就会将配置文件的路径初始化到kettle的全局变量中，所以需要在这个地方始做文章。

kettle初始化方法:

KettleEnvironment.init(false);

这个方法 默认一个boolean 类型的参数，用于判断是否要在此时初始化jndi 数据源，在另一篇文章

已有说明，他默认是true的，这里我给其传入false ，阻止他在此时初始化，将初始化的方法自己重
。

kettle 默认的初始化方法：

```
public static void initJNDI() throws KettleException {
   String path = Const.JNDI_DIRECTORY;

   if ( path == null || path.equals( "" ) ) {
     try {
       File file = new File( "simple-jndi" );
       path = file.getCanonicalPath();
     } catch ( Exception e ) {
       throw new KettleException( "Error initializing JNDI", e );
     }
     Const.JNDI_DIRECTORY = path;
   }

   System.setProperty( "java.naming.factory.initial", "org.osjava.sj.SimpleContextFactory" );
   System.setProperty( "org.osjava.sj.root", path );
   System.setProperty( "org.osjava.sj.delimiter", "/" );
 }
```

其实可以看出来，核心的一行代码其实是：

```
System.setProperty( "org.osjava.sj.root", path );
```

我猜，在kettle运行的时候，就是去获取了 org.osjava.sj.root 的值来获取数据源的，

那么，既然我阻止了他自己初始化jndi，而我又要使用jndi ，就要自己去初始化了：

```
/**
    * 初始化 jndi 数据源
    * @param kettleParams 参数
    */
   static void initJNDI(KettleParams kettleParams) {

       logger.info("正在初始化jndi数据源.....");
       try {
           Properties properties = new JndiProperRead().properties;

           OutputStream fos = new FileOutputStream(JndiProperRead.iniPath + "jdbc.properties"
;

           // 为避免自动转义，不用Properties自带的store方法。
           BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos, StandardChars
ts.UTF_8));

           // 先读取原本配置的ODS层固定数据源
           bw.write("############ODS JNDI数据源############");
           bw.newLine();
           for(Enumeration<?> e = properties.keys(); e.hasMoreElements();) {
               String key = (String)e.nextElement();
               String val = properties.getProperty(key);
               bw.write(key + "=" + val);
```

```
                bw.newLine();
            }
            bw.newLine();
            // 动态配置SRC 数据源信息。
            bw.write("############SRC JNDI数据源############");
            bw.newLine();
            bw.write("SRC/type="+getDbType(kettleParams.getDbType()));
            bw.newLine();
            bw.write("SRC/driver=oracle.jdbc.driver.OracleDriver");
            bw.newLine();
            bw.write("SRC/url=oracle.jdbc.driver.OracleDriver");
            bw.newLine();
            bw.write("SRC/user=oracle.jdbc.driver.OracleDriver");
            bw.newLine();
            bw.write("SRC/password=oracle.jdbc.driver.OracleDriver");
            bw.flush();

            //初始化kettle jndi 数据源。
            System.setProperty( "java.naming.factory.initial", "org.osjava.sj.SimpleContextFactory" )

            System.setProperty( "org.osjava.sj.root", JndiProperRead.iniPath.replace("jdbc.properti
s","") );
            System.setProperty( "org.osjava.sj.delimiter", "/" );
        }catch (Exception e){
            e.printStackTrace();
        }
    }
```

在这个自定义的初始化方法中，我读取了配置文件，通过修改了配置文件的内容以后，重新保存的。

kettle 读取的时候就可以读取到最新的配置文件了。

说明

这里面我在修改、保存 properties 文件的时候，没有使用Properties 类，而是直接以文件流的形式改的，因为用Properties 的store 保存以后，我发现，“：” 会被添加转义符号，但是读取的时候又会去掉转义符号，会导致数据库连接不上的问题。