将kettle嵌入springboot web项目中-简单实现

# 目的：

具体来说，我用java写一个springboot的项目，这个项目集成pentaho pdi sdk，从而可以在服务器端手动或者定时运行用pdi spoon编写的transformation和job（转换和任务)。

kettle是一个用得很广泛的ETL工具，是pentaho pdi的一部分。pdi里面包括了kettle,spoon等等。为了统一称呼，我们将pentaho的包括kettle,spoone这一系列的工具集简称为pdi。

# 准备工作：

首先需要下载下pentaho官方提供的demo：

<https://github.com/pentaho/pdi-sdk-plugins>

这个仓库里有如何为pdi编写插件以及集成pdi-sdk的例子，我们只需要这个目录kettle-sdk-embedding-samples下的项目即可。

集成pdi sdk的文档可以看这里：

<https://help.hitachivantara.com/Documentation/Pentaho/8.2/Developer_Center/PDI/Embed>

另外我们需要下载pdi的最新的版本：

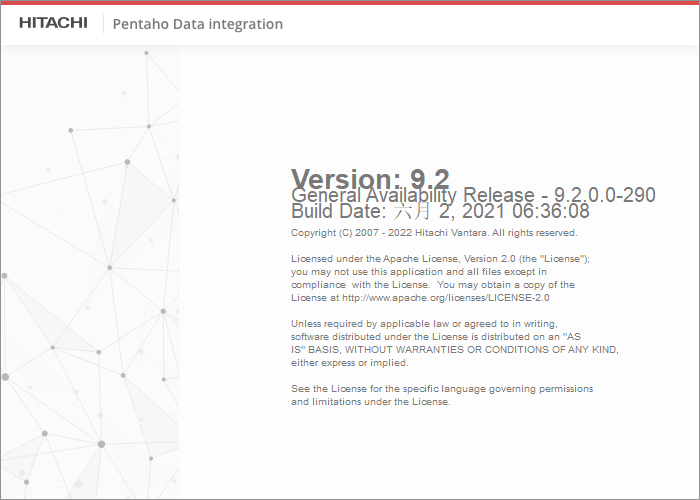
<https://sourceforge.net/projects/pentaho/>

目前我下载的是9.2.0.0-290版本，这个版本只能在JDK1.8下运行。

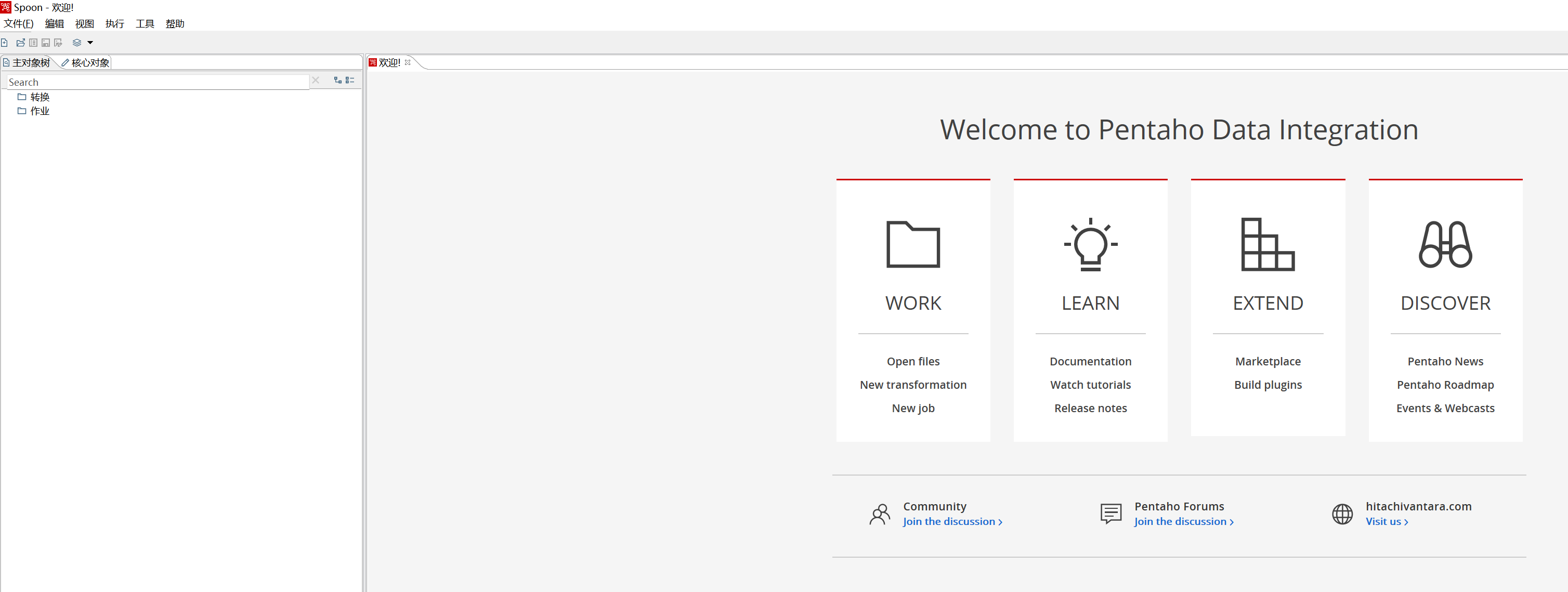
当然本机还需要准备好maven和idea。

# 用spoon创建转换demo

下载的pentaho pid 9.2.0.0-290,解压以后，双机spoon.bat(Windows 10 64位)即可直接运行。（记得缺省的JDK要选1.8版本)



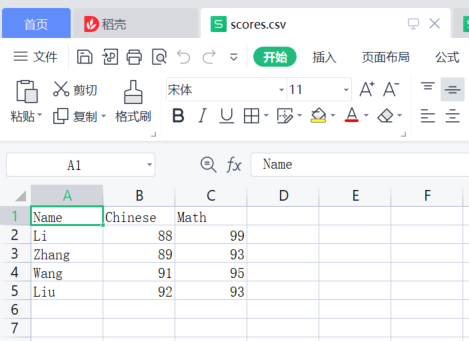
启动以后是这样的界面：



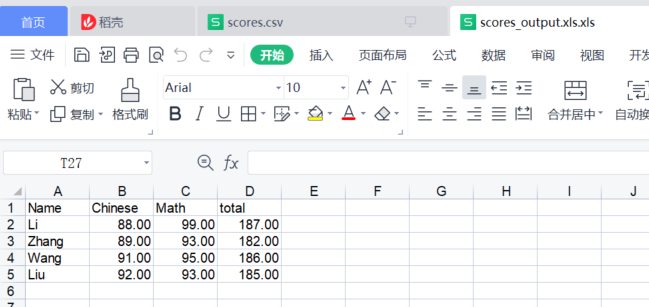
pdi可以创建转换和作业，作业可以看做多个转换的集合。

我们用pdi写一个简单的demo，这个demo不涉及数据库操作，只是简单的计算和转换。复杂的涉及数据操作的后面讲。

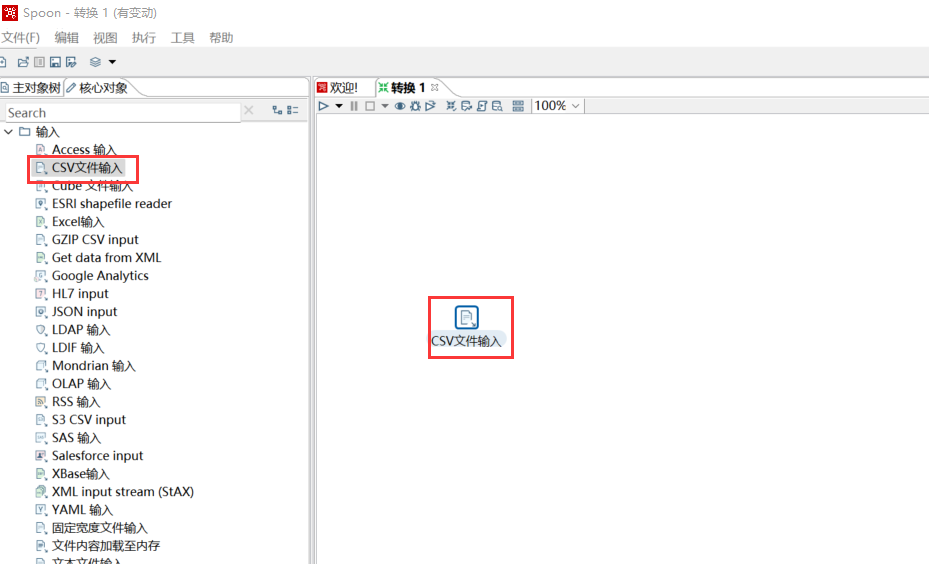
这个demo将下面这个scores.csv文件中：



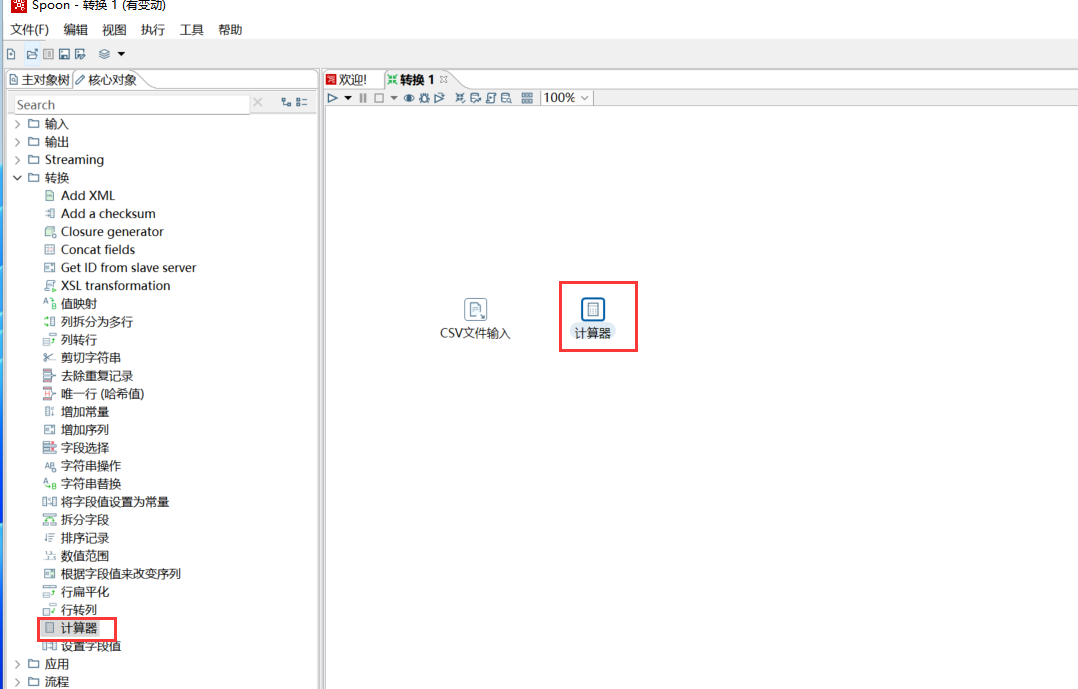
语文(Chinese),数学（Math)成绩加起来计算总分，记录在total这个字段，并输出到excel文件中，如下：



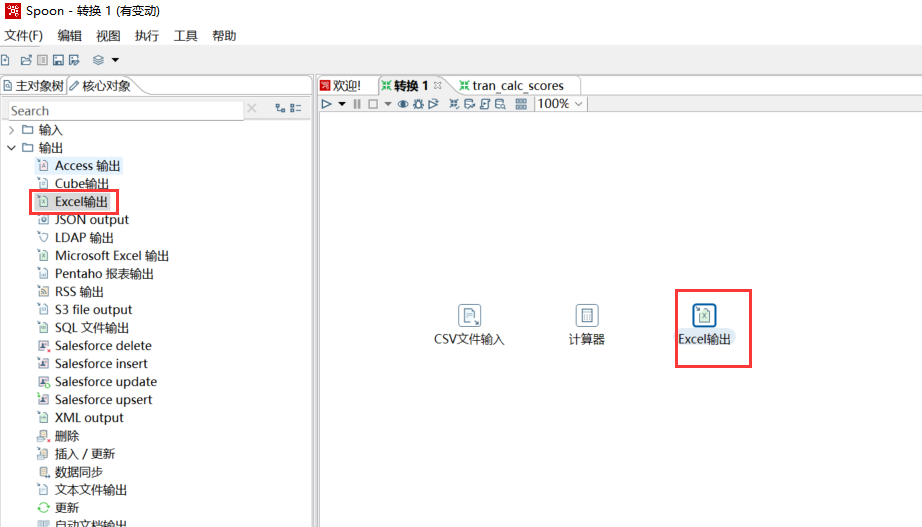
点菜单"文件"->"新建"->"转换"，从“输入分组”先拖一个"csv文件输入"对象到页面上：



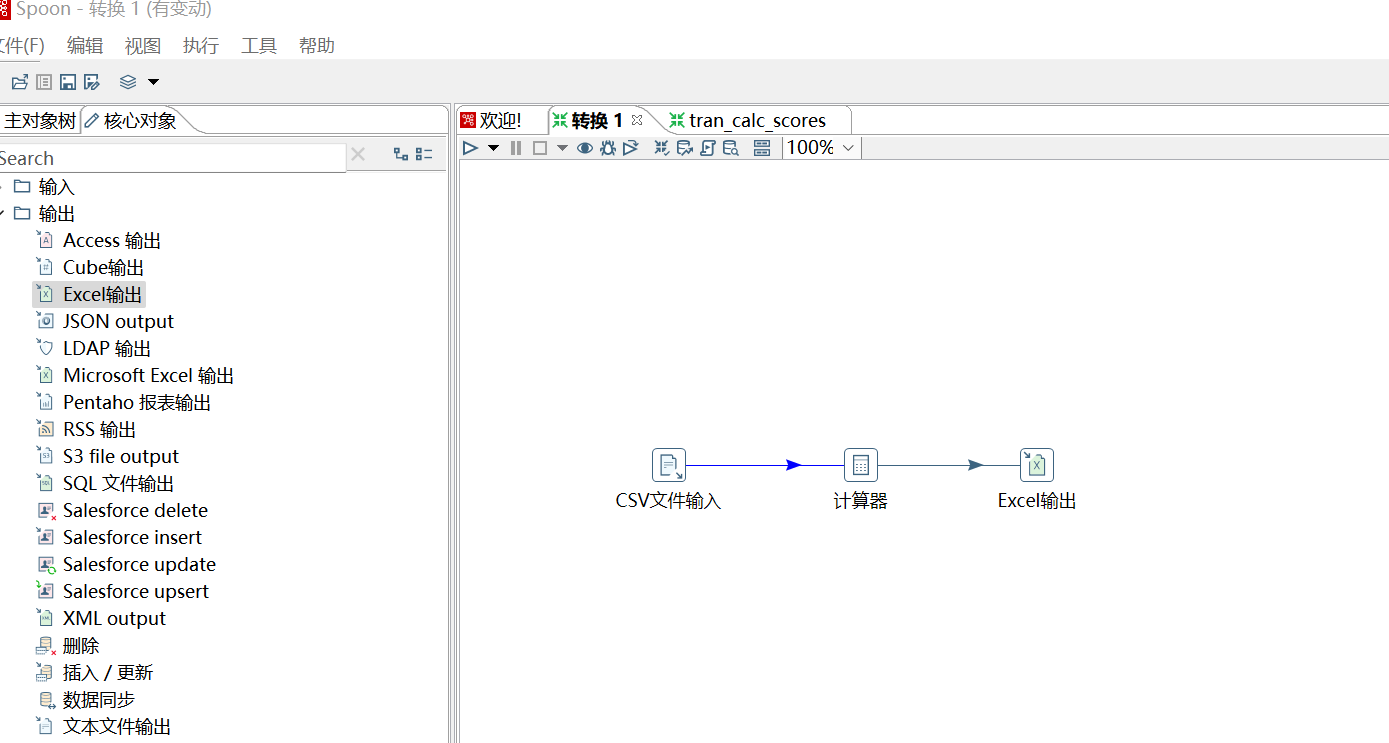
然后从“转换分组”拖一个“计算器”到页面上：



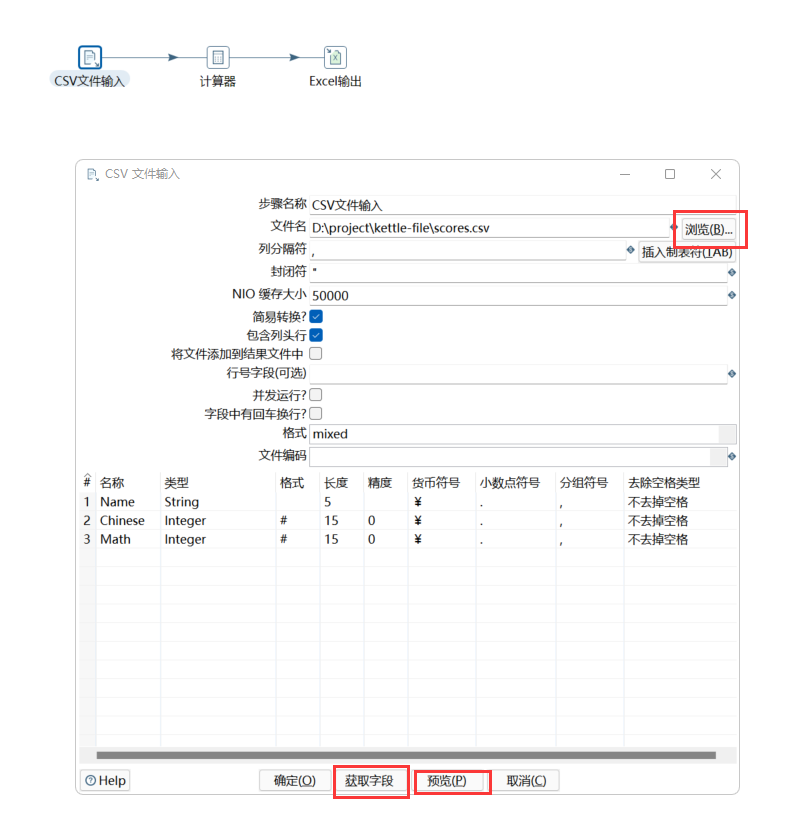
最后从“输出”分组拖一个“excel输出”到页面上：

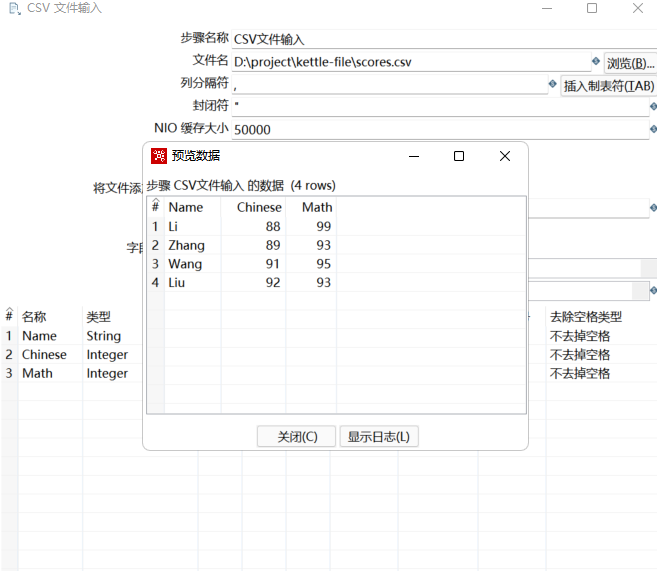


将几个对象按下面的顺序连接起来（按住shift键点鼠标拖连接线)



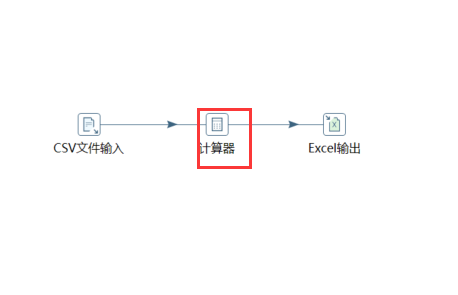
双击“csv文件输入”，点"浏览"选择上面提到的scores.csv文件(这个文件内容很简单，可以自己手工输入一下)，然后点"获取字段"查看一下字段，也可以点”预览"查看文件内容：



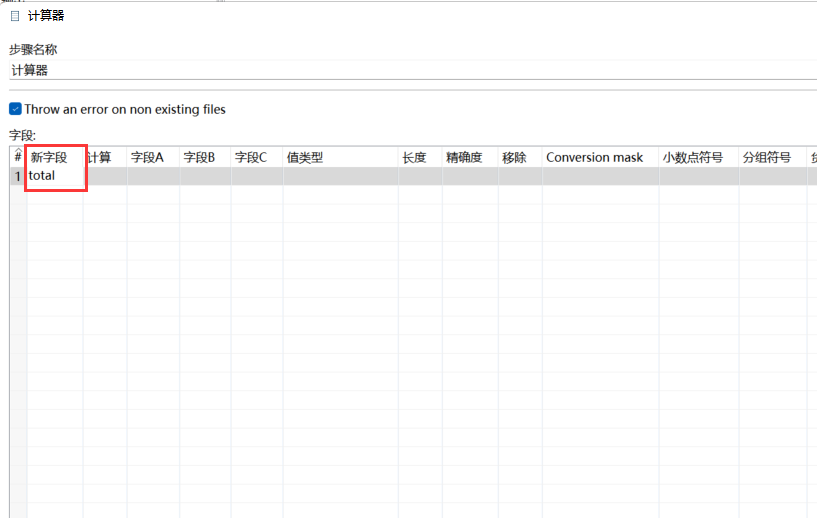


"csv输入文件"这个步骤我们就处理完了，接下来处理"计算器“步骤。这个步骤需要把Chinese和Math字段加起来，写入total。

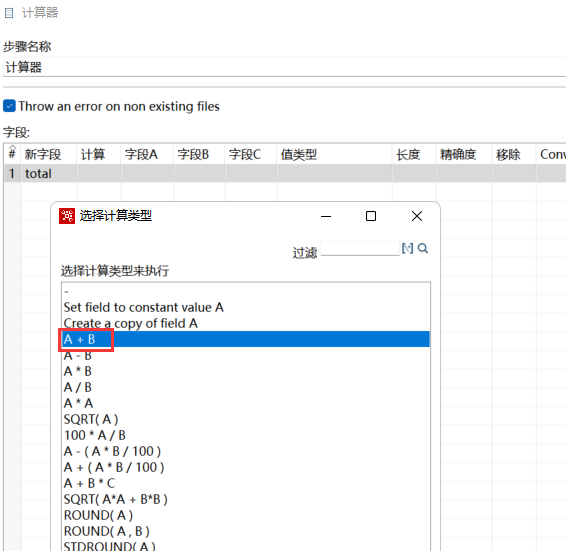
我们双击"计算器",

·

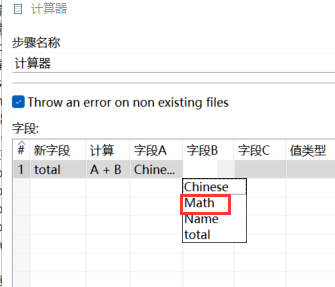
在下面"新字段"中填入total:



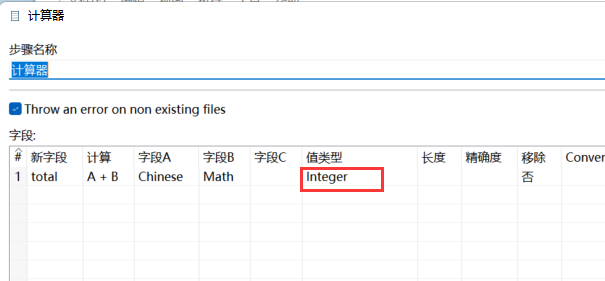
"计算"选"A+B"



字段选"Chinese"和”Math"：



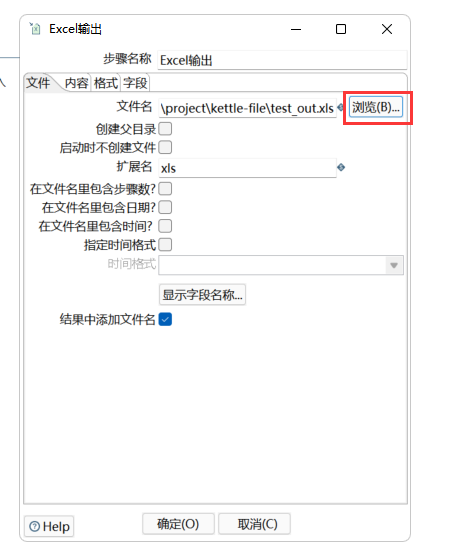
下面的“值类型”选Integer



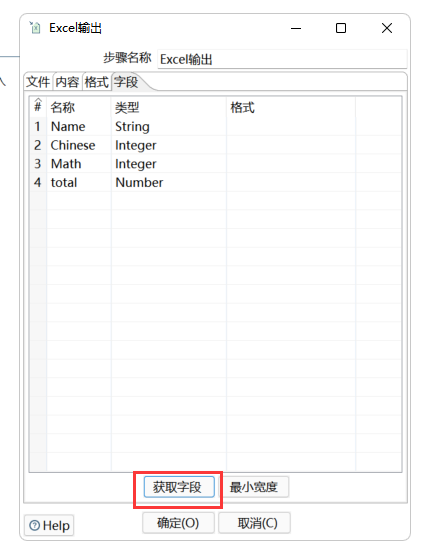
接下来双击"Excel输出"



点击“浏览”输入输出的文件名"test\_out.xls"

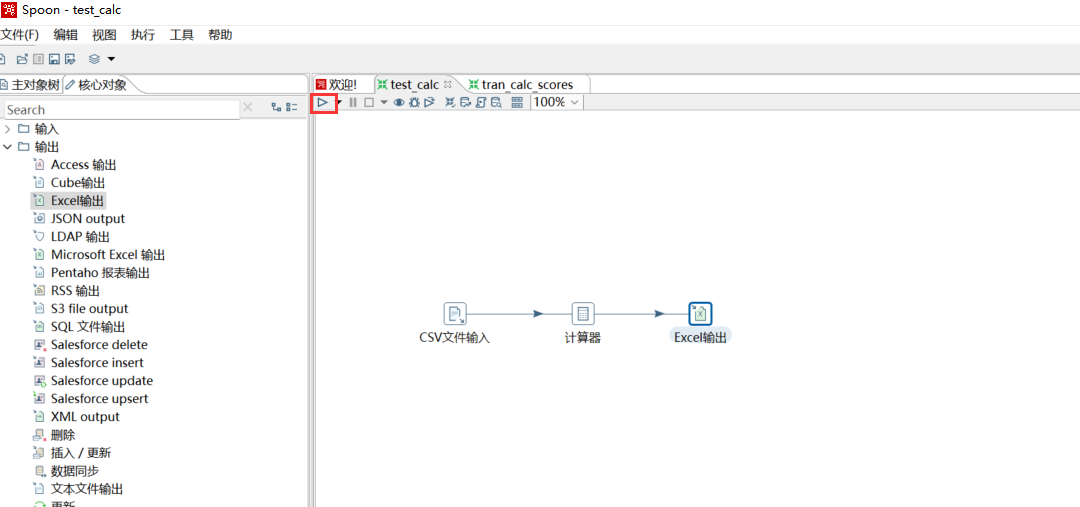


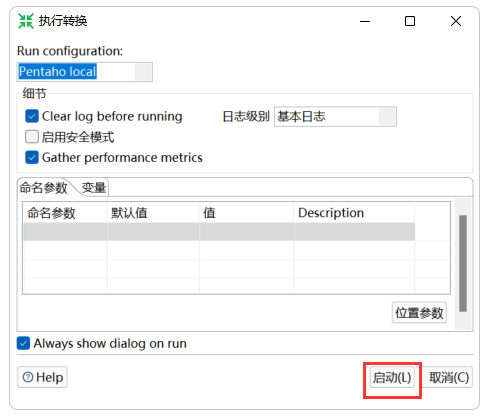
然后在“字段”页签上点"获取字段"



然后确定。

这个时候ctrl+s保存一下这个转换，然后执行：

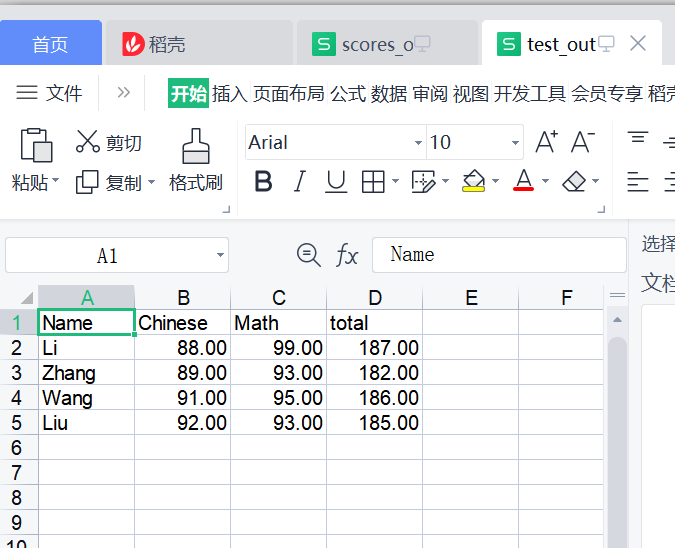




顺利的话出现下面这些输出：



检查test\_out.xls内容如下图，说明转换成功。

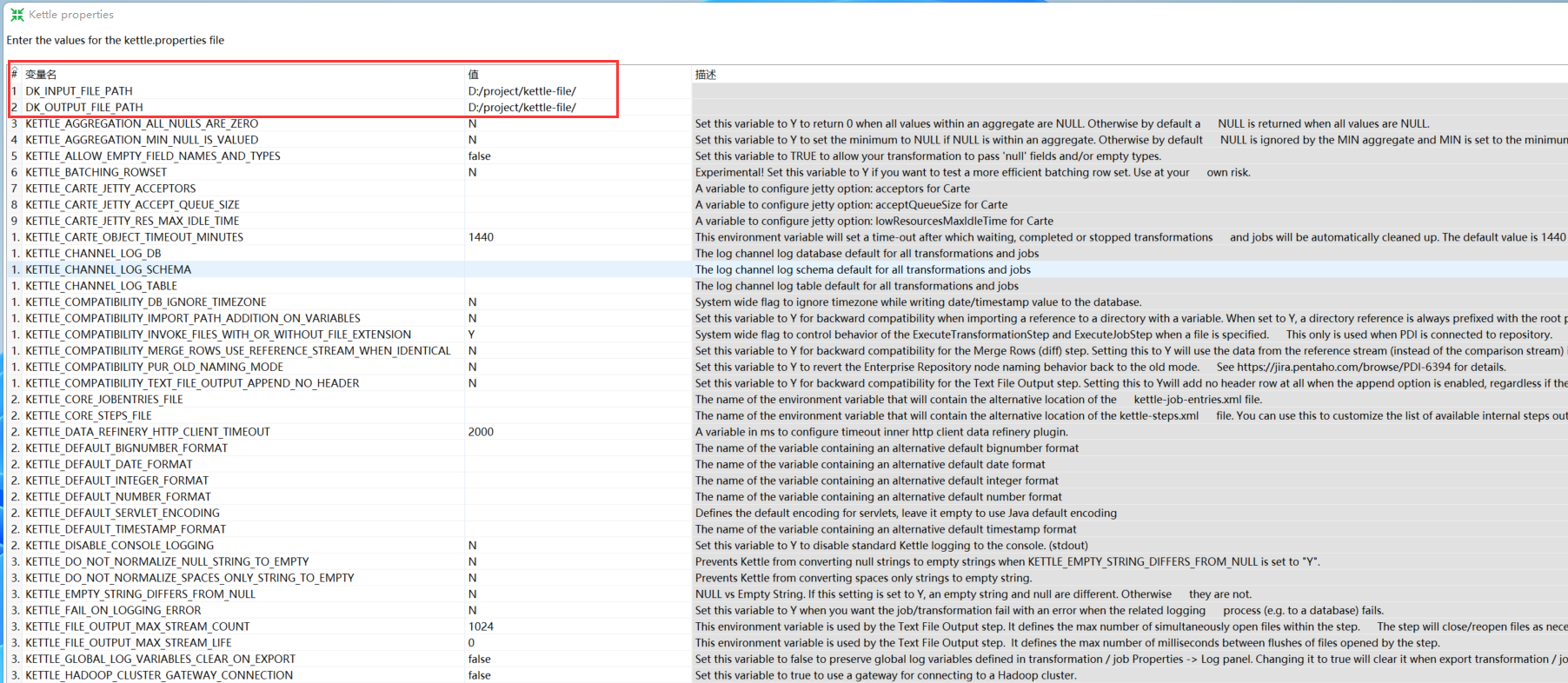


# 转换(tranformation)转移到服务器上运行的准备

根据上面的demo我们可以看出，一个pdi的转换存到硬盘上是一个后缀为ktr的文件，用文本编辑器打开ktr文件，发现其实是个xml文件：

我们观察一下这个xml文件，发现输入输出文件的路径是绝对路径（上面的红框)。如果我们的springboot 应用集成了kettle engine在服务器端运行，即使是Windows Server的服务器都不一定会存在D:\project\kettle-file这个路径的，更何况如果是Linux是肯定没有这个路径。所以我们需要定义两个常量，来表示输入文件的路径和输出文件路径。

spoon上的菜单 "编辑"->"编辑kettle.properties文件"，打开是如下界面：



添加如下两个常量并设置好路径：

DK\_INPUT\_FILE\_PATH = D:/project/kettle-file/

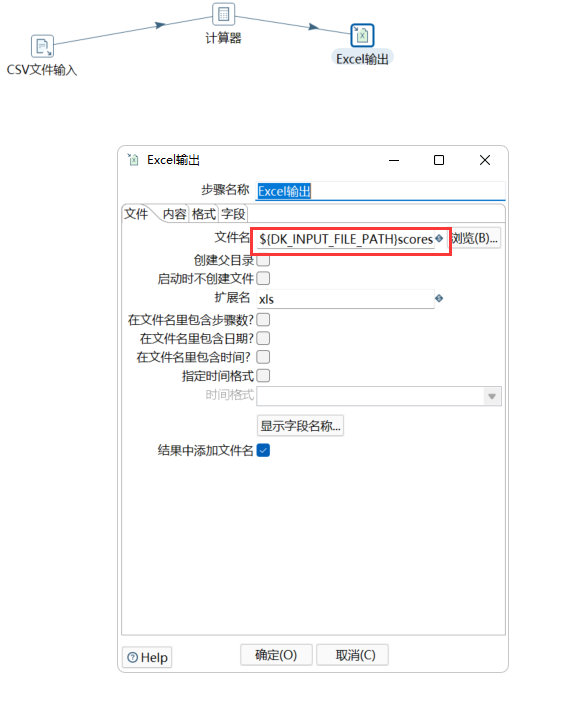
DK\_OUTPUT\_FILE\_PATH = D:/project/kettle-file/

也可以直接找到本机上的kettle.properties，用文本编辑器打开，添加这两个常量。



然后在spoon中修改输入文件和输出文件的路径,如下图:





即可。

# 创建maven项目

用idea创建一个空白的maven项目，pom.xml文件的内容可以直接从kettle-sdk-embedding-samples这个项目拷贝过来，但是需要修改几行。

这个pom.xml文件的parent设为：

版本需要改如上面的9.2.0.2-388。

这里之所以选pentaho-kettle做parent而不是org.springframework.boot，是因为springboot支持不用parent的方式创建springboot项目，详细可以看这里：

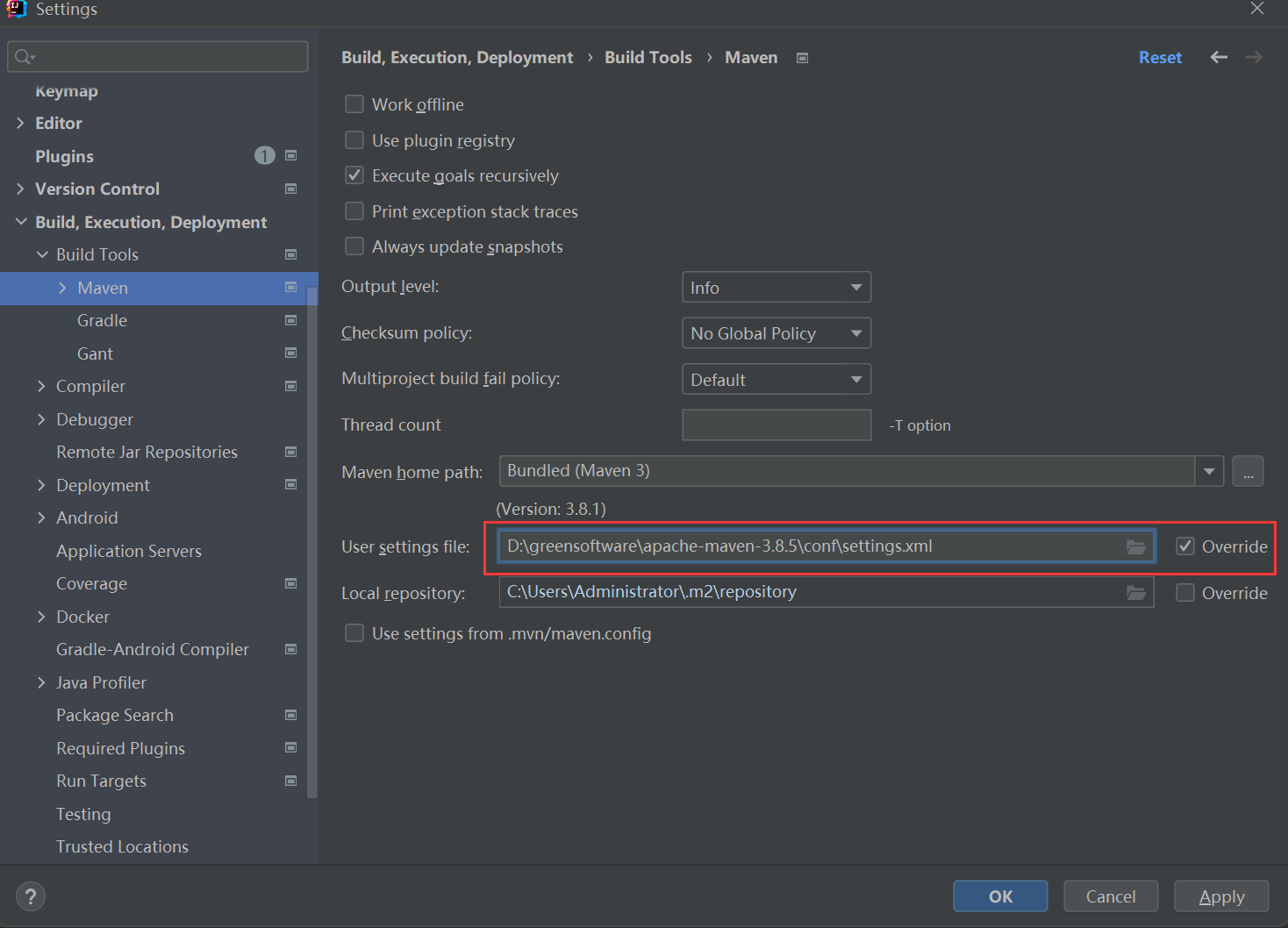
<https://docs.spring.io/spring-boot/docs/1.2.3.RELEASE/reference/html/using-boot-build-systems.html#using-boot-maven-without-a-parent>

完整的pom.xml文件如下：

上面这个pom.xml并不是一次性写好的，中间解决过如下的的编译和运行的错误：

另外maven的settings.xml文件需要更新，加上pentaho的仓库地址、阿里云的仓库地址等，完整的文件内容如下：

修改完settings.xml文件，在idea工程中使用这个文件：



# 关键代码kettleController.java如下：

以上代码中，关键就如下两点:

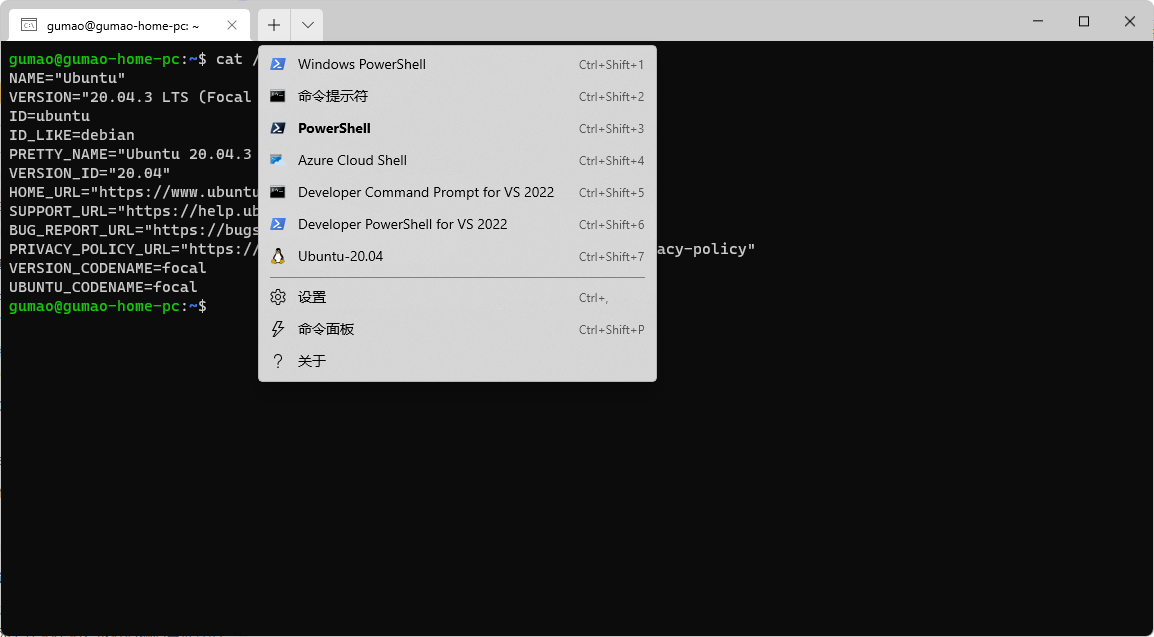
1.在类的构造函数中把kettle运行环境初始化，如下：

2.在KettleController的 /kettle/runtrans接口响应方法中运行对应的ktr文件：

代码写好解决完编译错误就可以用maven package命令打包。

# 运行环境配置

我是在Windows 11的wsl2的Ubuntu 20.4中运行demo的(Windows 10最新的版本也支持wsl2).



首先需要在Ubuntu中配置kettle.properties文件，这个文件放在

~/.kettle/目录下，kettle.properties就如下两行：

DK\_INPUT\_FILE\_PATH = /home/gumao/kettle-file/

DK\_OUTPUT\_FILE\_PATH = /home/gumao/kettle-file/

然后修改一下application.yml文件,内容如下：

然后可以开始运行，运行前检查一下java的版本是JDK1.8

运行：

java -jar dkservice-1.0-SNAPSHOT.jar

# 解决运行问题/异常

运行前检查一下java的版本是JDK1.8

运行：

java -jar dkservice-1.0-SNAPSHOT.jar

第一次出现如下错误：

|  |
| --- |
| Caused by: java.lang.NoSuchMethodError: 'java.lang.String javax.servlet.ServletContext.getVirtualServerName()'  at org.apache.catalina.authenticator.AuthenticatorBase.startInternal(AuthenticatorBase.java:1319) ~[classes!/:1.0-SNAPSHOT]  at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183) ~[classes!/:1.0-SNAPSHOT]  ... 48 common frames omitted |

这个是因为依赖冲突，运行：

mvn -U dependency:tree -Dverbose

检查依赖树，发现依赖了2个javax.servlet:javax.servlet-api:的版本：

行 81: [INFO] | | +- javax.servlet:servlet-api:jar:2.4:compile

行 82: [INFO] | | +- javax.servlet:jsp-api:jar:2.0:compile

行 191: [INFO] +- javax.servlet:javax.servlet-api:jar:3.1.0:test

行 191: [INFO] +- javax.servlet:javax.servlet-api:jar:3.1.0:test

在dependency中排除掉就好了。

|  |
| --- |
| <dependency>  <groupId>pentaho-kettle</groupId>  <artifactId>kettle-engine</artifactId>  <version>${pdi.version}</version>  <exclusions>  <exclusion>  <artifactId>servlet-api</artifactId>  <groupId>javax.servlet</groupId>  </exclusion>  </exclusions>  </dependency> |

解决了上面这个还有类似的问题：

|  |
| --- |
| ... 64 more  Caused by: java.lang.NoClassDefFoundError: com/fasterxml/jackson/databind/ser/std/ToStringSerializerBase  at java.base/java.lang.ClassLoader.defineClass1(Native Method)  at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:1012)  at java.base/java.security.SecureClassLoader.defineClass(SecureClassLoader.java:150)  at java.base/java.net.URLClassLoader.defineClass(URLClassLoader.java:524)  at java.base/java.net.URLClassLoader$1.run(URLClassLoader.java:427)  at java.base/java.net.URLClassLoader$1.run(URLClassLoader.java:421)  at java.base/java.security.AccessController.doPrivileged(AccessController.java:712)  at java.base/java.net.URLClassLoader.findClass(URLClassLoader.java:420)  at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:587)  at org.springframework.boot.loader.LaunchedURLClassLoader.loadClass(LaunchedURLClassLoader.java:151)  at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:520)  at java.base/java.lang.Class.getDeclaredConstructors0(Native Method)  at java.base/java.lang.Class.privateGetDeclaredConstructors(Class.java:3373)  at java.base/java.lang.Class.getConstructor0(Class.java:3578)  at java.base/java.lang.Class.getDeclaredConstructor(Class.java:2754)  at org.springframework.beans.BeanUtils.instantiateClass(BeanUtils.java:147)  ... 77 more  Caused by: java.lang.ClassNotFoundException: com.fasterxml.jackson.databind.ser.std.ToStringSerializerBase  at java.base/java.net.URLClassLoader.findClass(URLClassLoader.java:445)  at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:587)  at org.springframework.boot.loader.LaunchedURLClassLoader.loadClass(LaunchedURLClassLoader.java:151)  at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:520)  ... 93 more |

上面这个问题，应该是应该是对com.fasterxml.jackson这个包有依赖冲突，在项目的pom.xml中加入依赖：

这样会短路掉其它的冲突的依赖项，以这个最近的依赖项为准。

解决了这个问题，运行如下命令

java -jar .\dkservice-1.0-SNAPSHOT.jar

输出两行卡死在

# Initializing Spring root WebApplicationContext.

然后就想加 -debug调试一下，运行：

java -jar .\dkservice-1.0-SNAPSHOT.jar -debug

但是输出如下错误：

解决上面这个问题，首先需要用everything找到本机的spring-boot-autoconfigure-2.4.5.jar文件，用解压缩软件打开之后，将meta-info/spring.factories文件拷贝到项目对应的目录下。