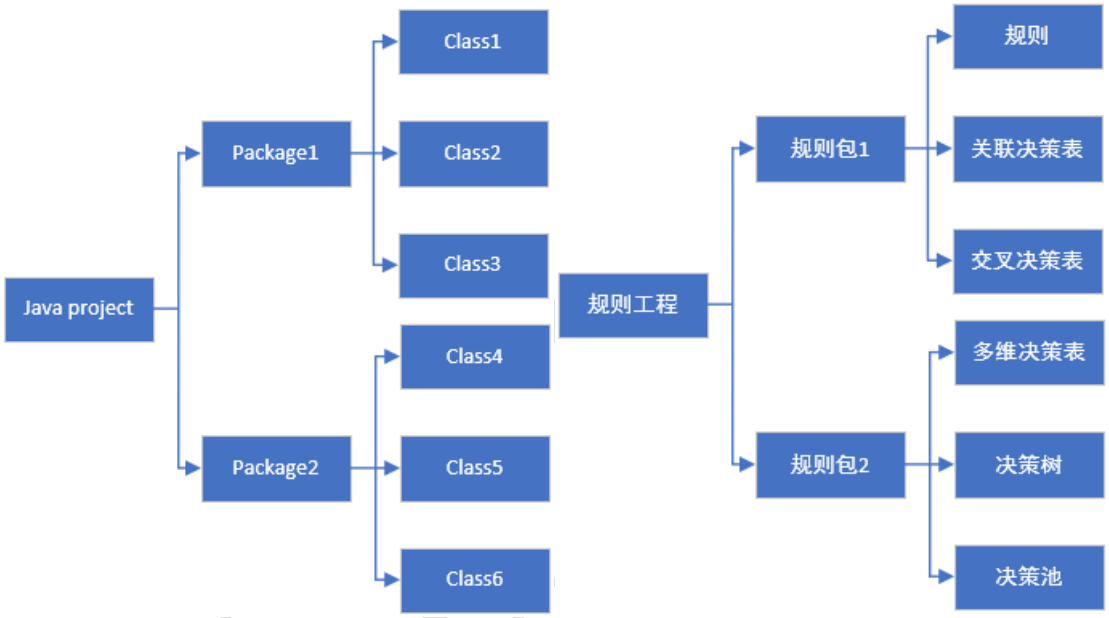
规则配置器操作手册

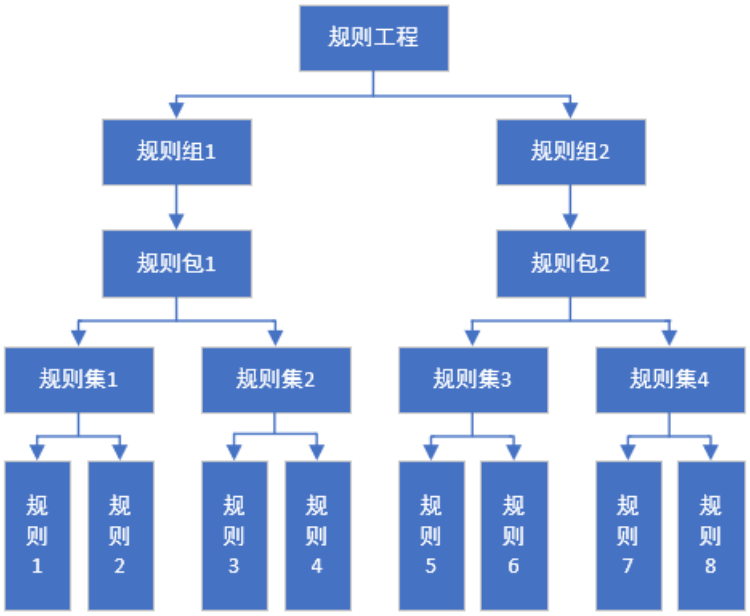
目 录

一、 规则配置器简介

使用规则配置器创建项目与用代码创建项目类似，例如在 Java 中我们可以新建一个“Java project”，而在规则配置器里我们则把它称之为“规则工程”。与“Java project”创建“package”相同，在“规则工程”中我们也可以创建若干“规则包”，不同的是这些“规则包”即可以单独运行，也可以相互调用组成一个整体。而在“规则包”下的就是具体的“规则”了，这等价于“package”下的“class”。在规则引擎中规则是最小的单位，与之同级的还有“关联决策表”、“交叉决策表”、“多维决策表”、“决策树”、“决策池”等。



在“规则工程”中，相同类型的“规则包”我们可以把它放在同一“规则组”里，相同作用的“规则”可以放在同一“规则集”下。



在规则工程里，我们将规则中要用到的变量统一定义在“对象库”里，而“对象库”又分为“传入数据”、“临时数据”、“外部调用”和“常量数据”这四个模块。

其中“传入数据”是指：将规则运行所需数据通过赋值给“传入数据”中的变量来拿到规则中进行计算，再将结果值回传给“传入数据”中的某个变量，即传入与传出；

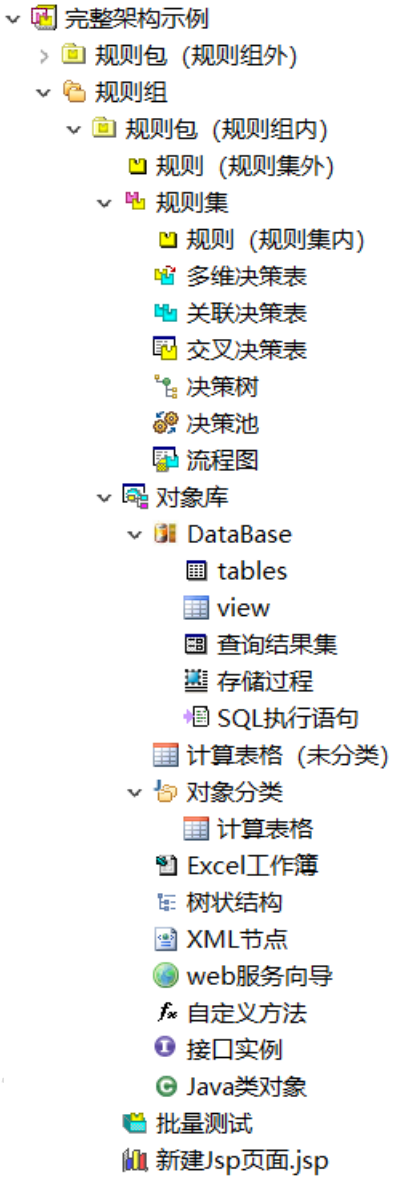
“临时数据”是指：在规则的计算过程中要用到的，而结果中不需要的，这样的数据我们把它定义在“临时数据”中；

“外部调用”是指：规则引擎将一些常用的方法进行了封装，在“外部调用”中通过“添加公式”添加此方法，在规则中直接调用此方法名即可引用；

“常量数据”是指：如果某个变量需要赋予常量值，则这个值我们就可以将其添加到常量数据中，通过常量数据赋给此变量。

在“对象库”下，可以建立数据库连接以获取数据库数据，以及“计算表格”、“树状结构”、“XML 节点”、“Java 类对象”、“自定义方法”等。

规则完成之后，要检测“规则包”的正确性，我们可以通过添加“批量测试”来进行测试。而如果需要页面则可直接添加一个“JSP 页面”，再经过简单的配置就可使用了。以下是“规则工程”架构示例图：



下面通过“Hello World”案例的开发，来体验规则引擎开发的具体流程：

1.1. 案例描述

本案例的需求是：

规则包：新建一个 HelloWorld 规则包，此规则包接收一个参数（用户姓名），如果姓名不为空，则返回欢迎词“Hello”+姓名，如果姓名为空，则返回欢迎词“Hello World”。

页面调用：通过规则配置器集成的页面配置器制作一个录入页面，在页面上输入姓名，点击【提交】按钮后，后台获取用户输入的姓名，将它传递给 HelloWorld 规则包并执行，将执行后的结果的欢迎辞显示在页面上。

1.2. 规则开发

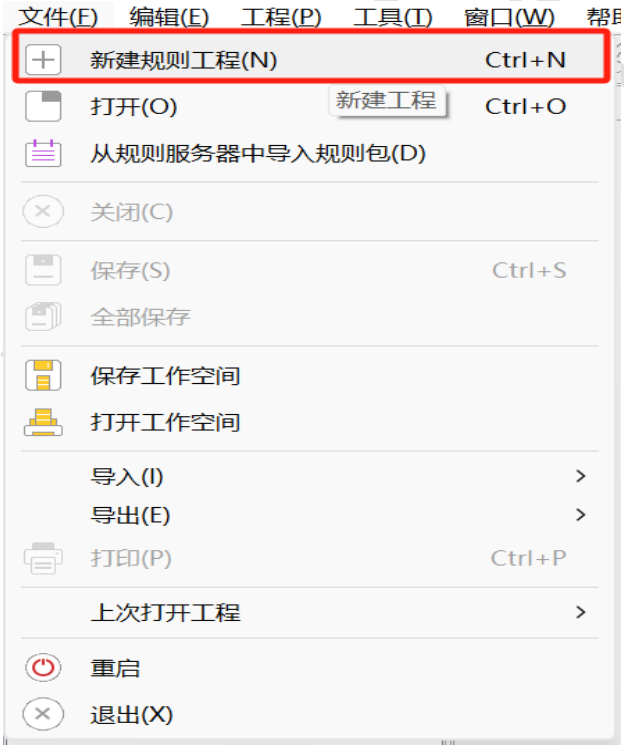
启动规则配置器。启动后，进入到缺省的开发工作空间。

规则配置器开发规则步骤包括创建工程、创建规则包、定义对象库、添加规则、发布规则包、测试规则包、创建 web 页面、web 方式测试规则包。

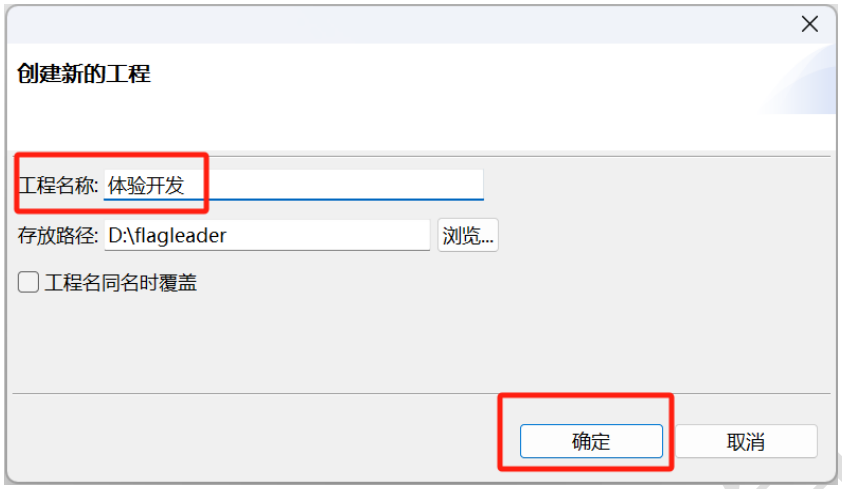
以下分别讲述这些步骤具体的操作方法：

1．创建工程

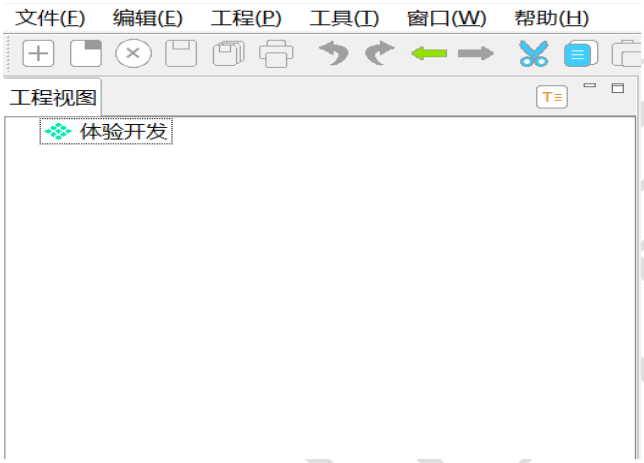
点击菜单栏上的“文件”，选择菜单项中的“新建规则工程”，如下图：



弹出对话框“创建新的工程”，在工程名称对应的文本框中输入“体验开发”。如果要更改存放路径点击“浏览...”，选择自己的存放路径，如下图：



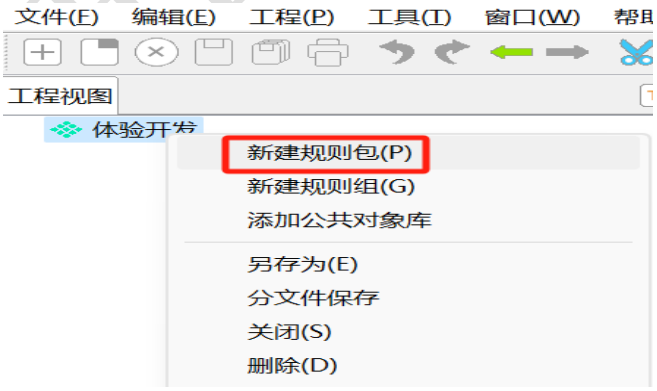
输入完成点击确定，如下图：



规则工程创建完成。

2. 创建规则包

右键“体验开发”规则工程，在菜单栏中选择“新建规则包”：



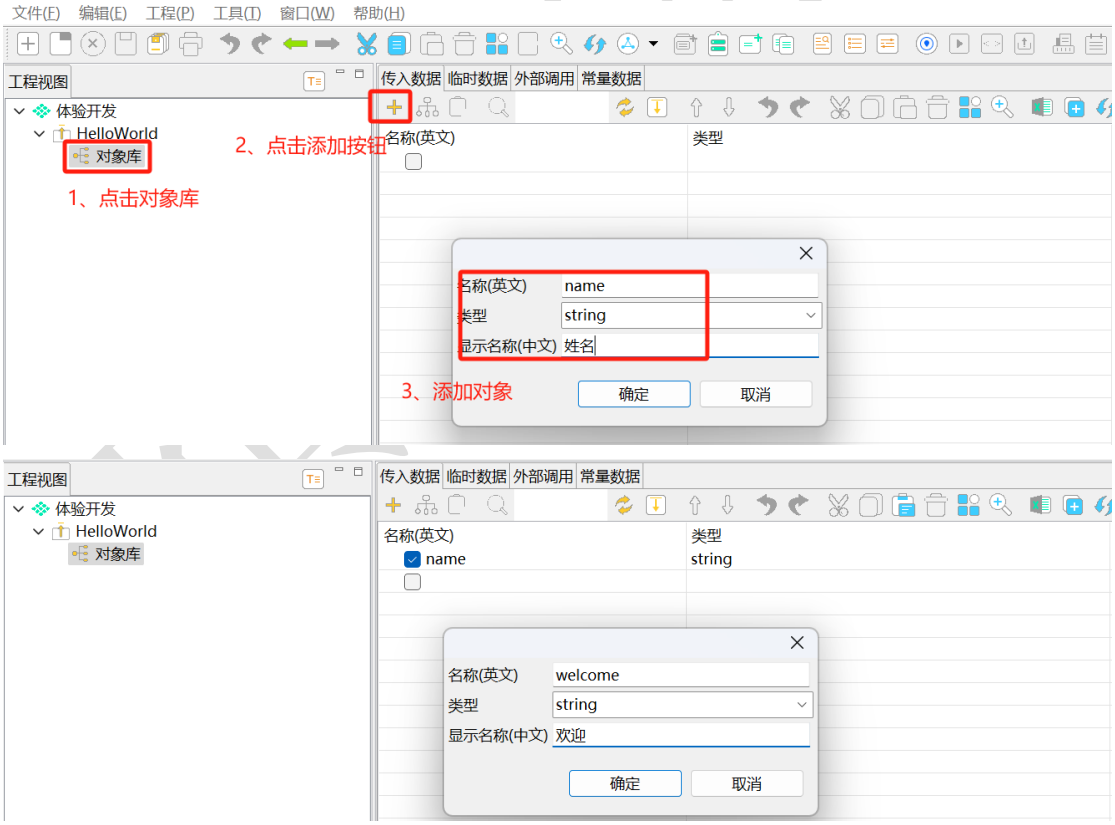
并将其命名为“HelloWorld”：

规则包创建完成。

3．定义对象库

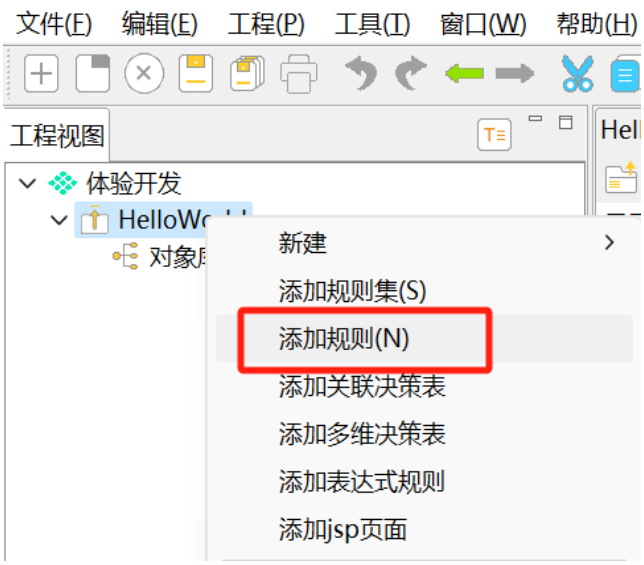
打开规则包“HelloWorld”，在对象库的传入数据中定义变量“姓名（name）”、“欢迎辞（welcome）”。

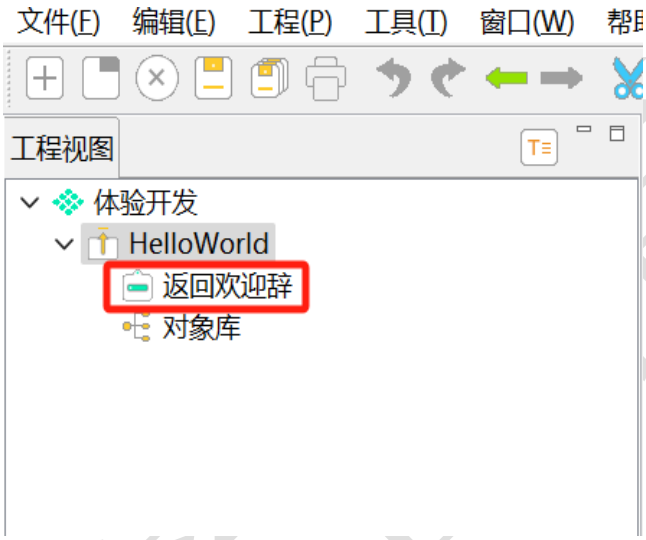
在编辑窗口的工具栏中点击“”添加变量，在弹出的对话框里依次填入名称、类型、显示名称。（名称必须为英文，类型默认为 String，这里根据自己的需求去在下拉框中选择，显示名称则可以为中文。）



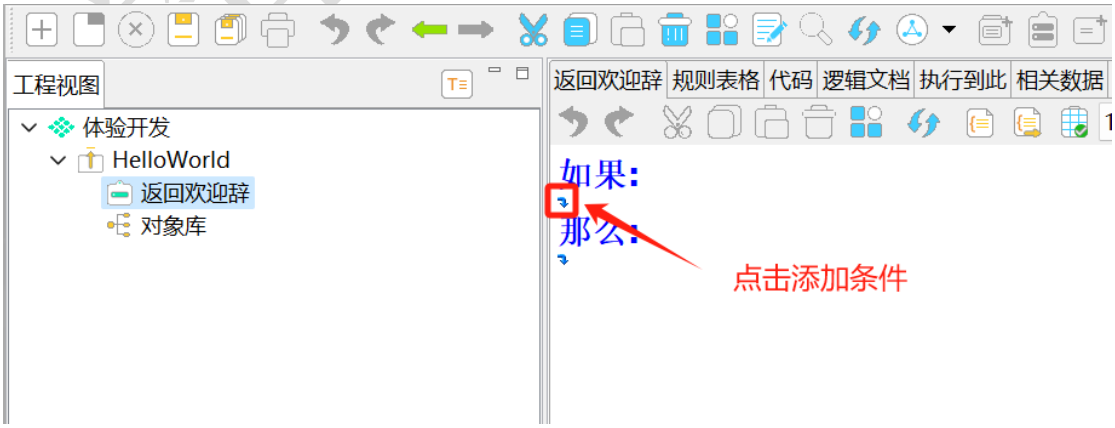
4．添加规则

选择规则包“HelloWorld”，右键“添加规则”，并将其命名为“返回欢迎辞”：

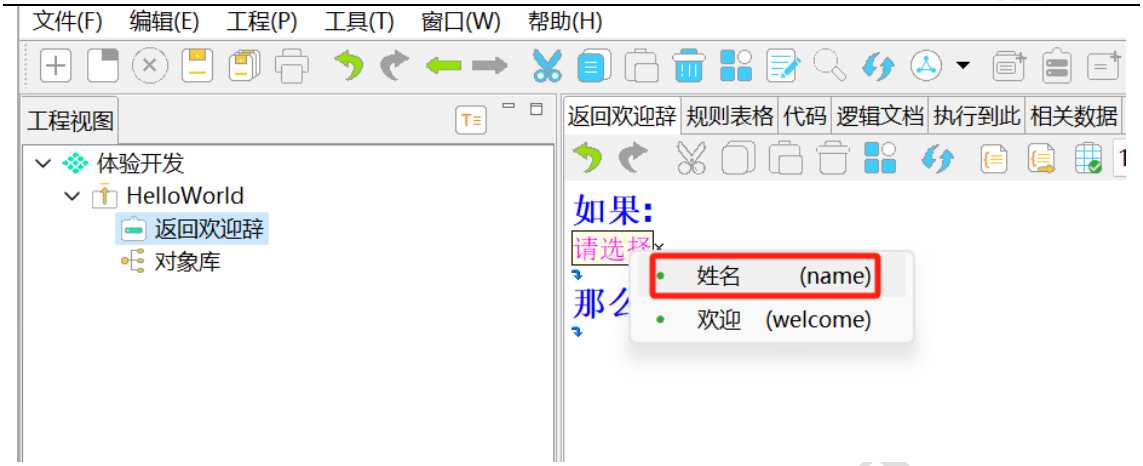




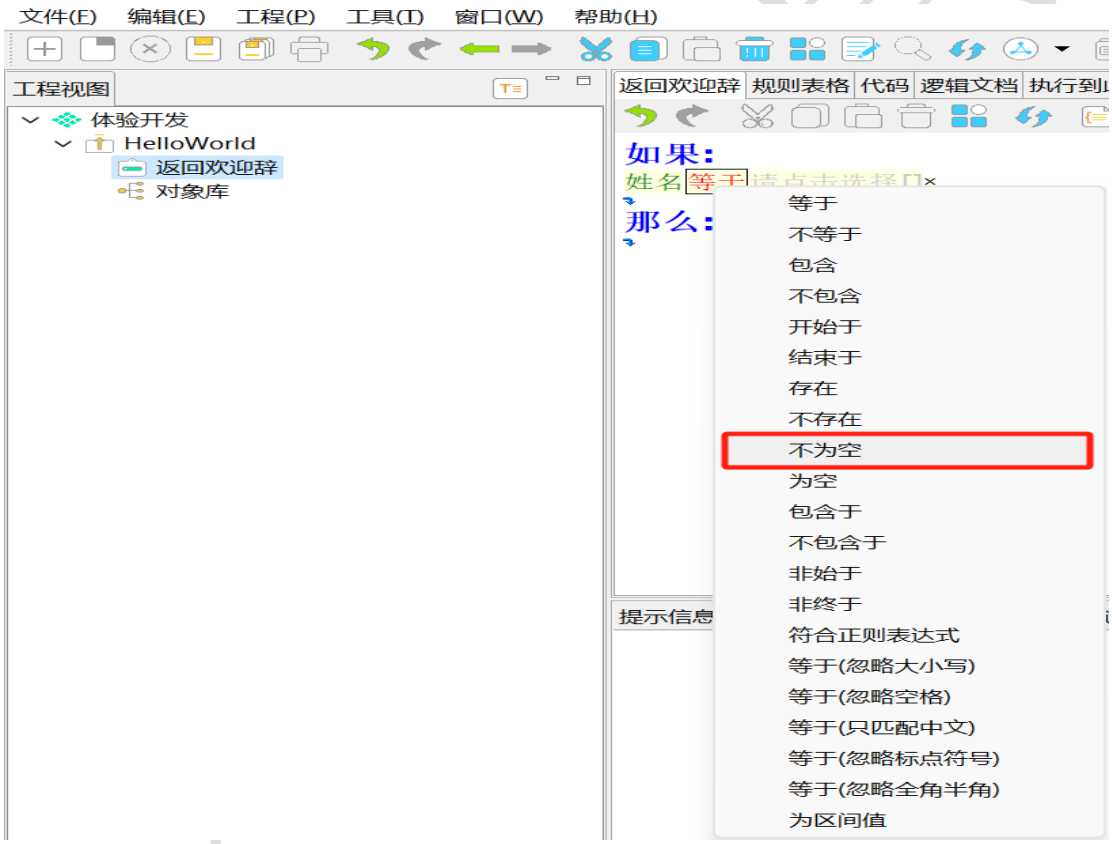
点击“返回欢迎辞”规则，在中间编辑窗口中，点击“如果”下面的“”添加条件：



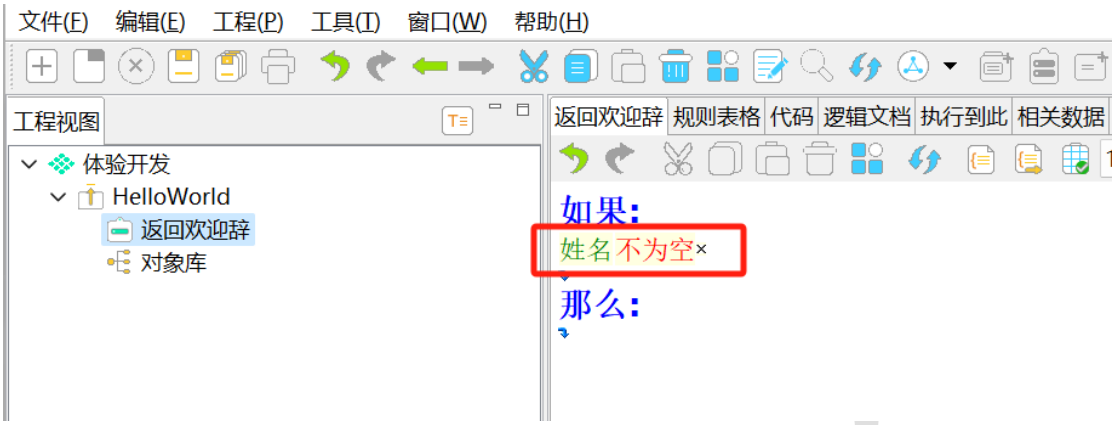
点击“”出现“请选择”，选择“姓名”：



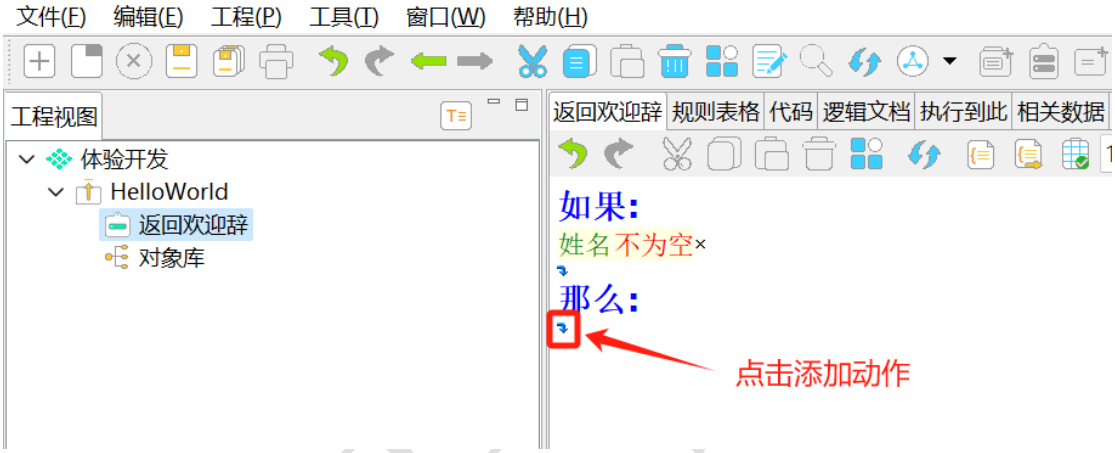
点击“姓名”后面的“等于”，出现菜单项，如下：



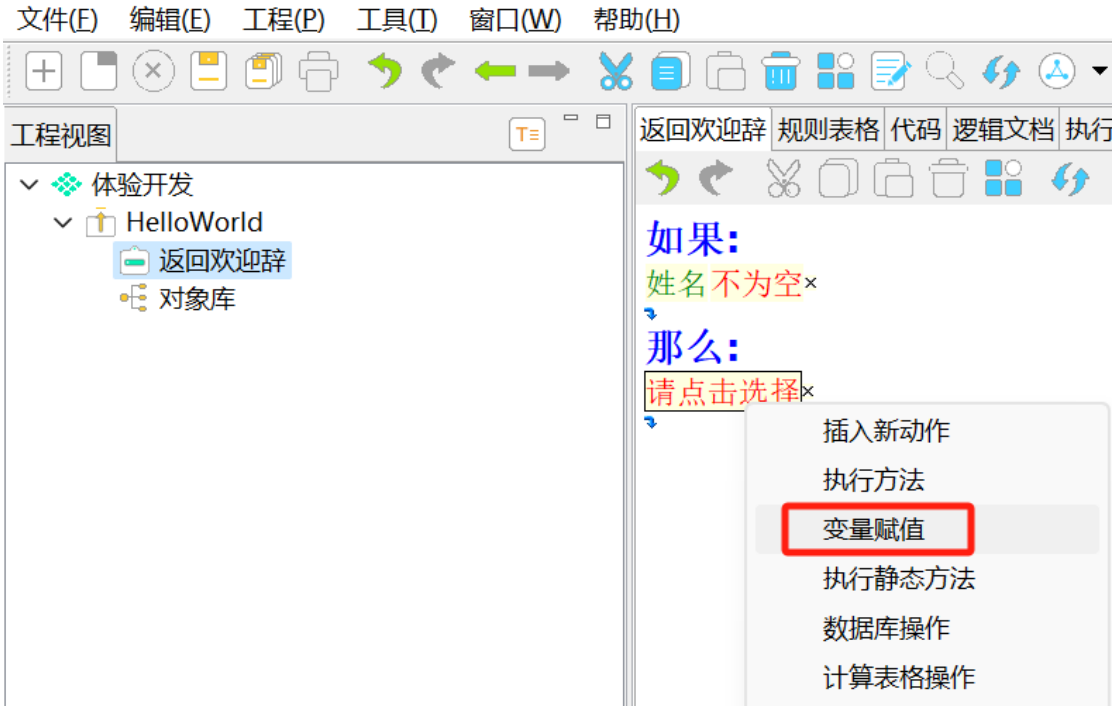
根据变量“姓名”类型的不同，点击“等于”出现在菜单项中的选项也会有所不同，根据需求选择相应值。这里姓名设置的类型为 String，需要判断“姓名”不为空。因此，将“等于”改为“不为空”：



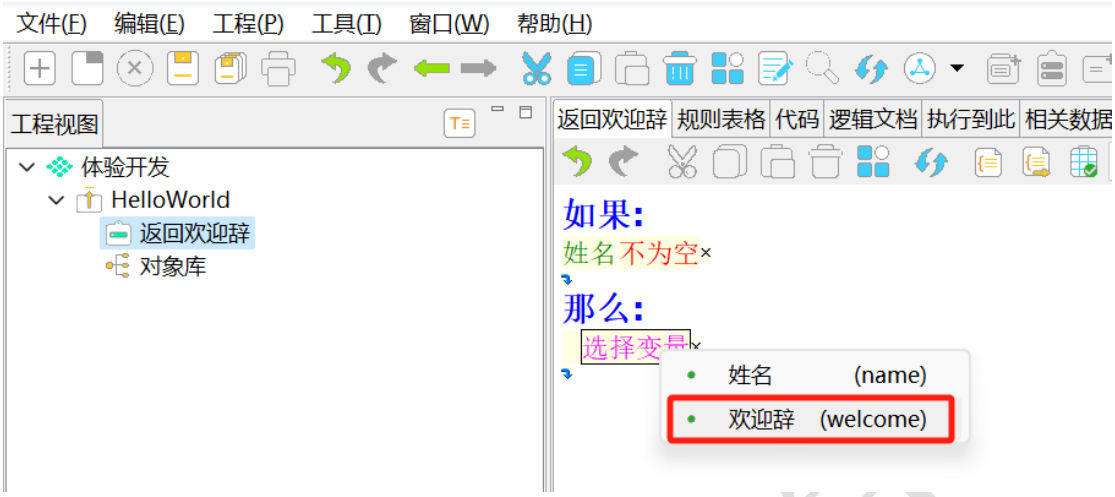
点击“那么”下面的“”添加动作：



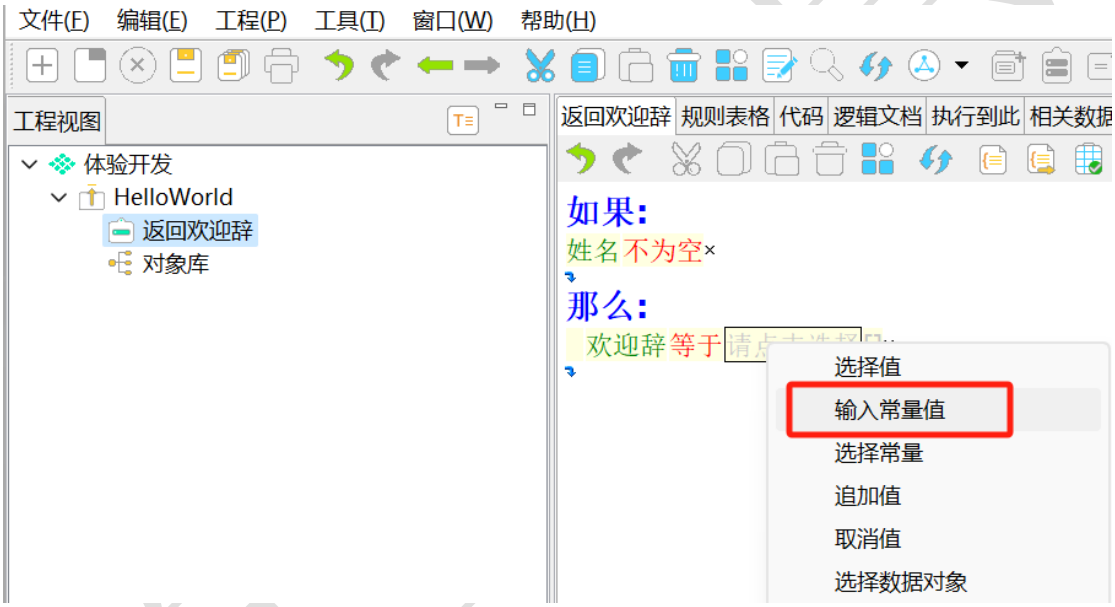
点击“”出现“点击请选择”，点击选择“变量赋值”：



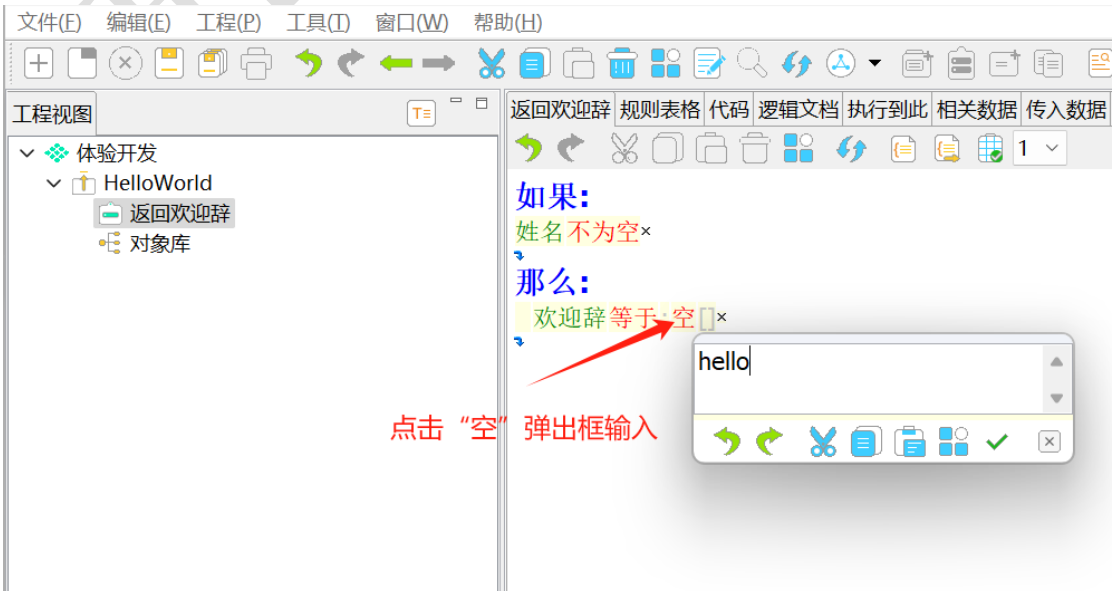
点击变量赋值，选择“欢迎辞”：



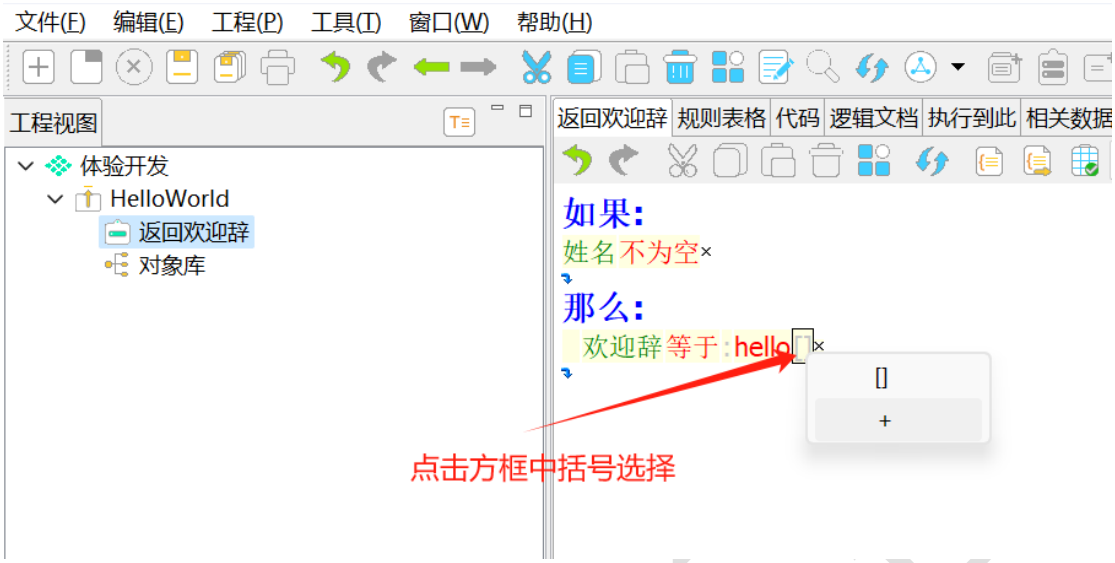
给变量“欢迎辞”添加常量值。点击“点击请选择”选择“输入常量值”：



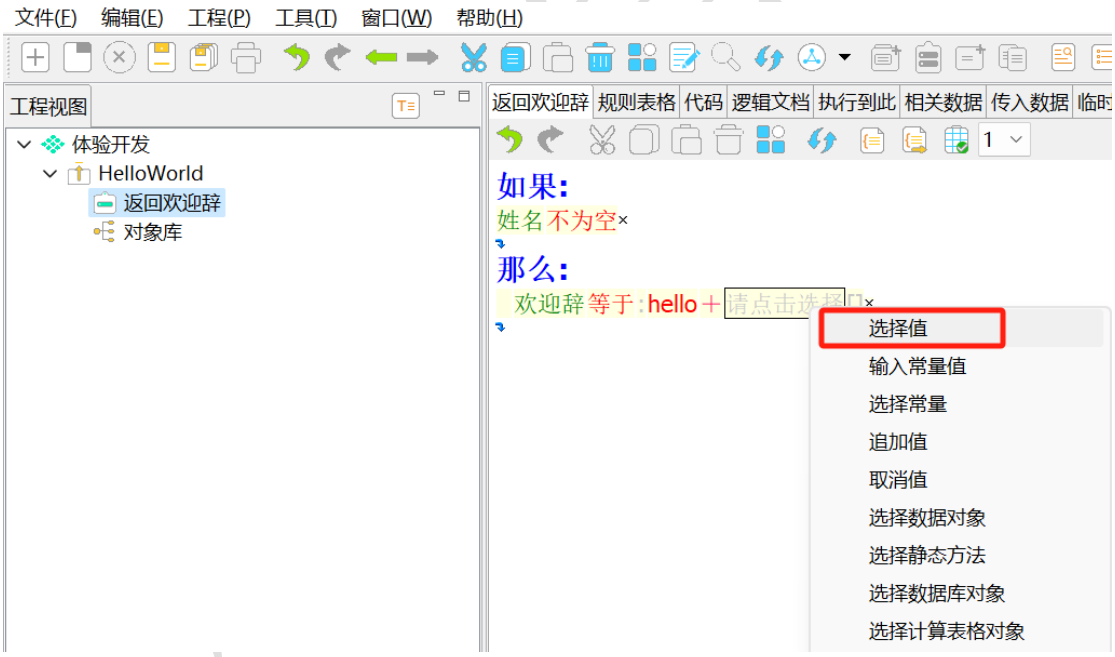
点击“空”，在弹出框中常量值“Hello ”：



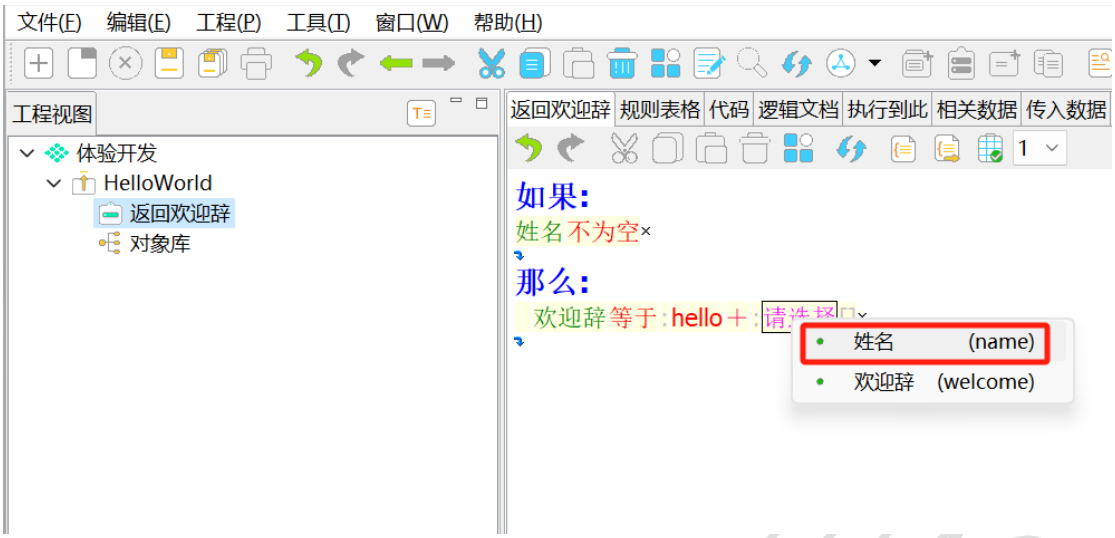
点击常量值“Hello ”后面的“[]”字符串间操作符添加运算符：



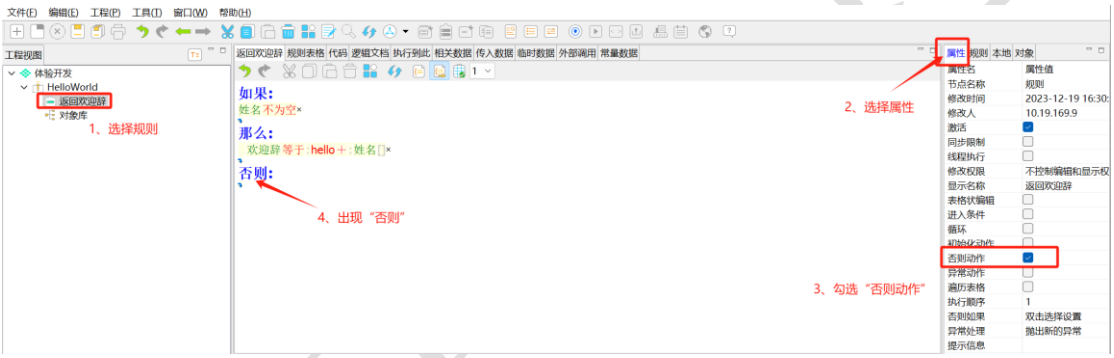
由于变量“欢迎辞”是 String 类型，所以这里字符串间操作符只有“+”。选择“+”后面出现“点击请选择”点击选择“选择值”：



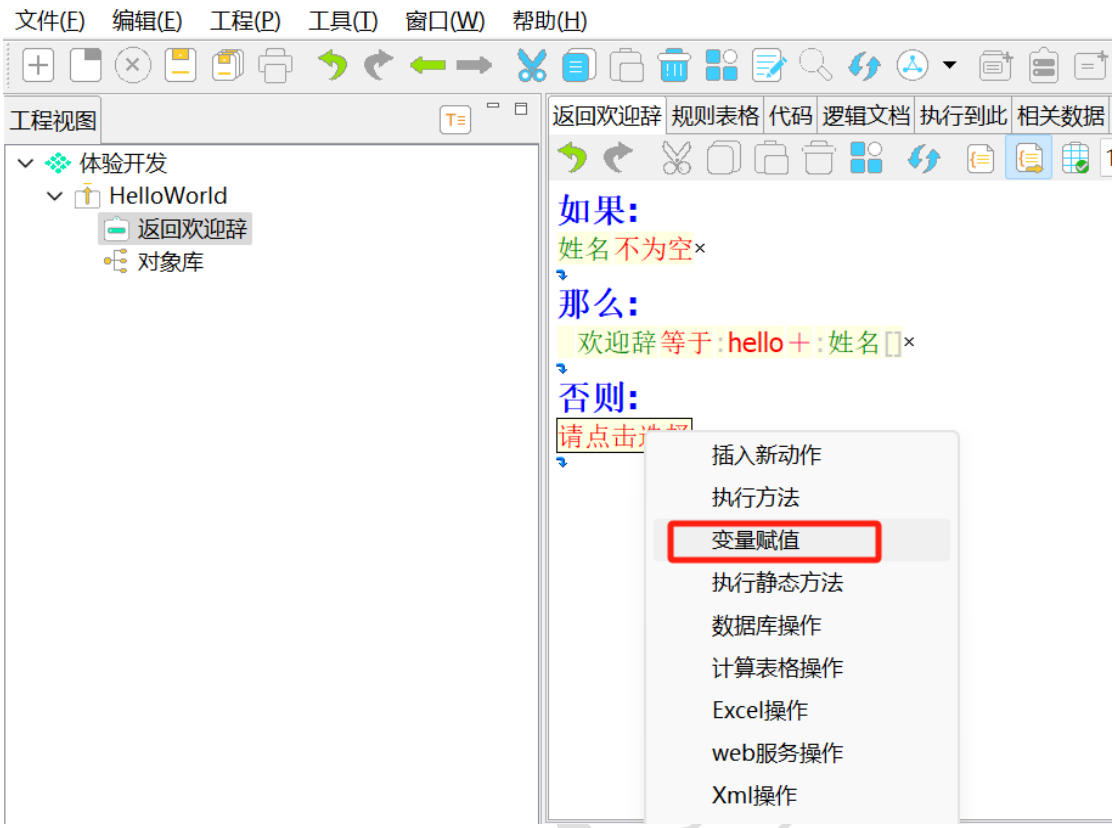
选择变量“姓名”：



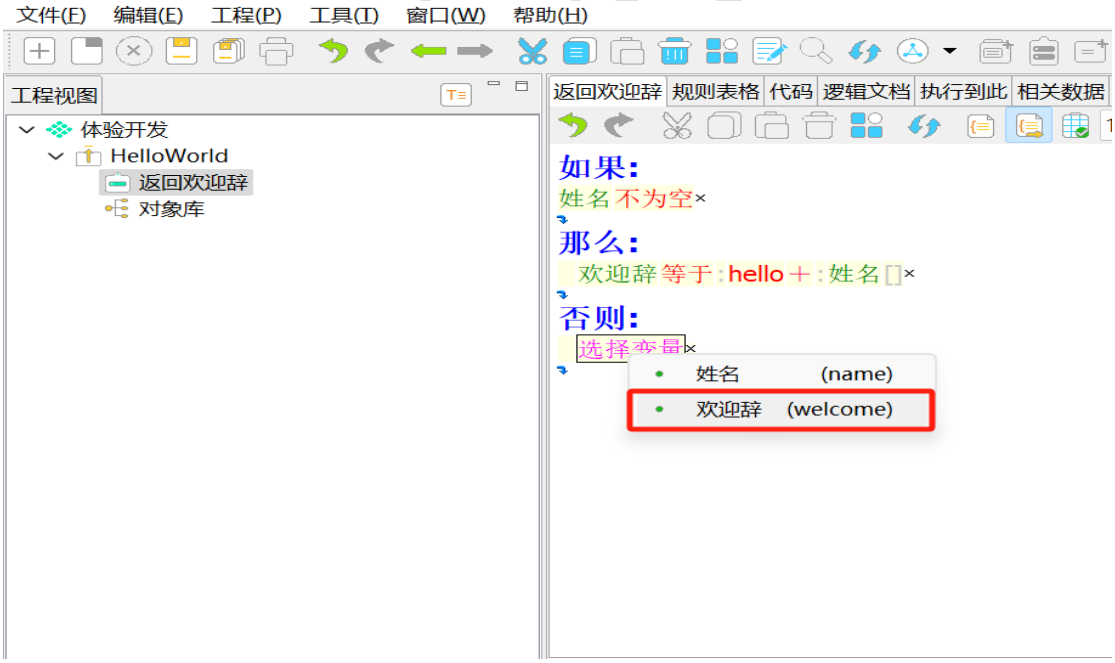
添加一个否则动作。在右侧属性窗口中，将属性名为“否则动作”的属性值一栏中的复选框勾选：



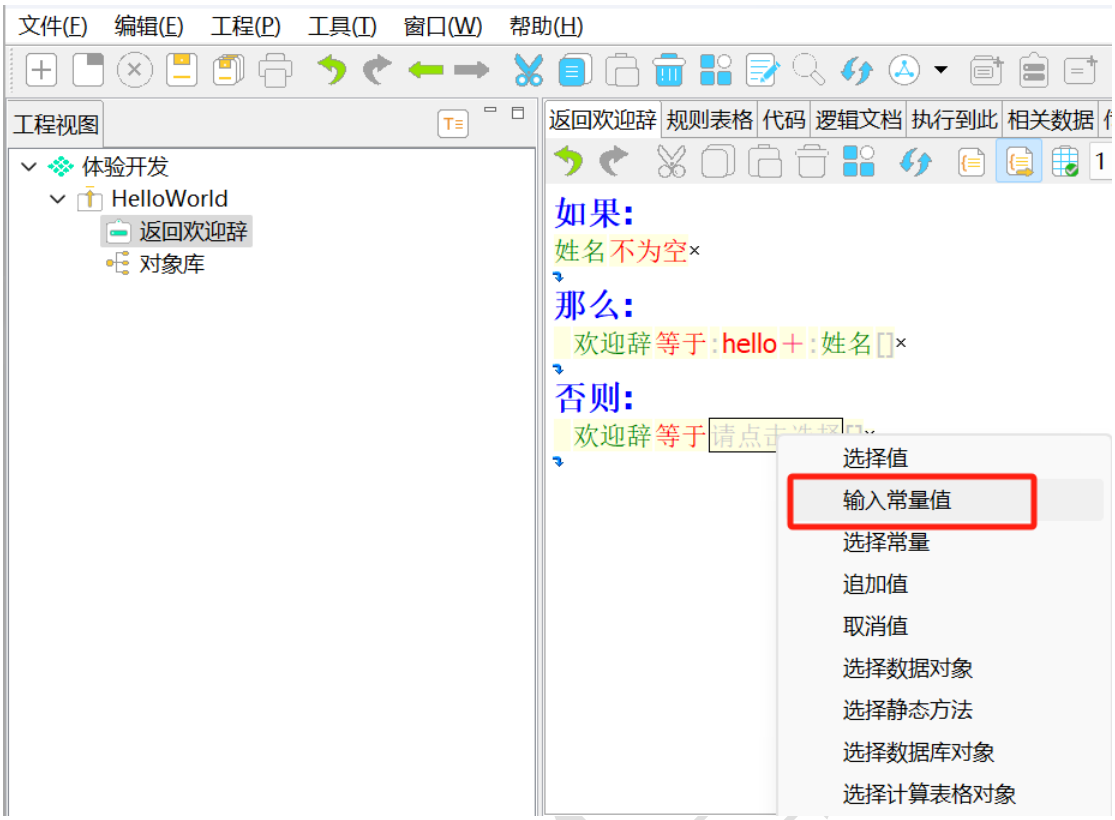
点击“否则”下面的“”，选择“变量赋值”：



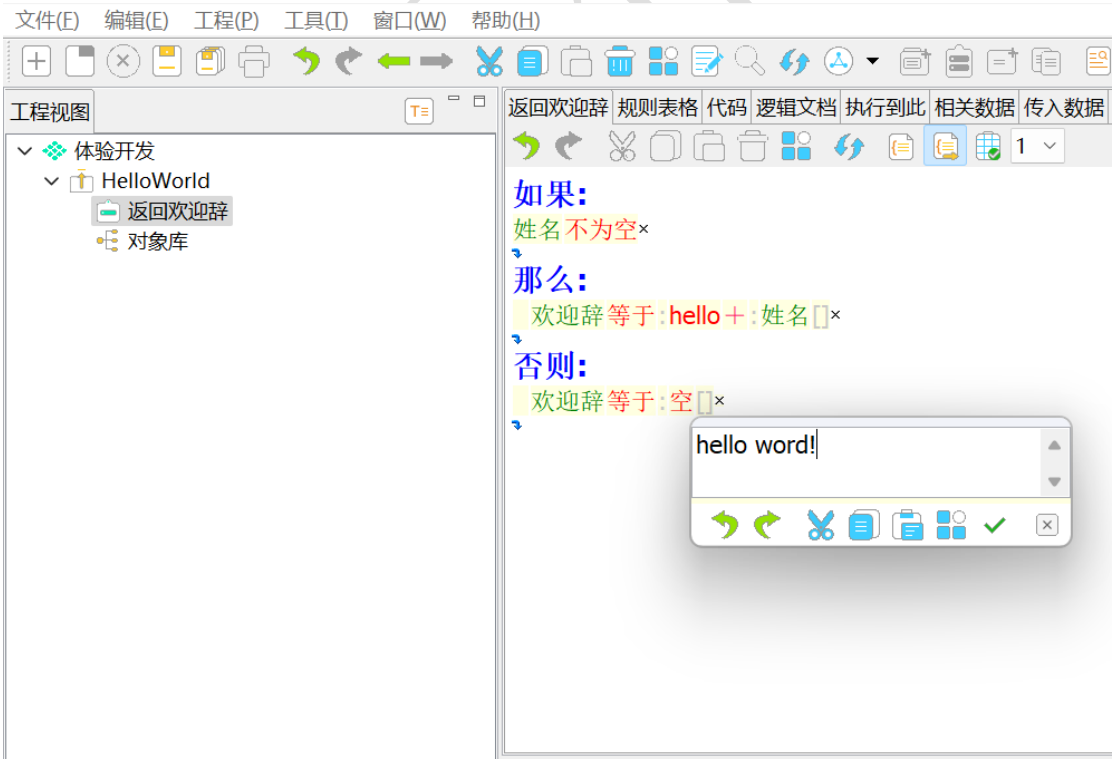
选择变量“欢迎辞”：



并给变量“欢迎辞”赋予常量值“Hello World”。点击“点击请选择”，选择“输入常量值”：



点击“空”，在弹出框中输入常量值：



5．发布规则包

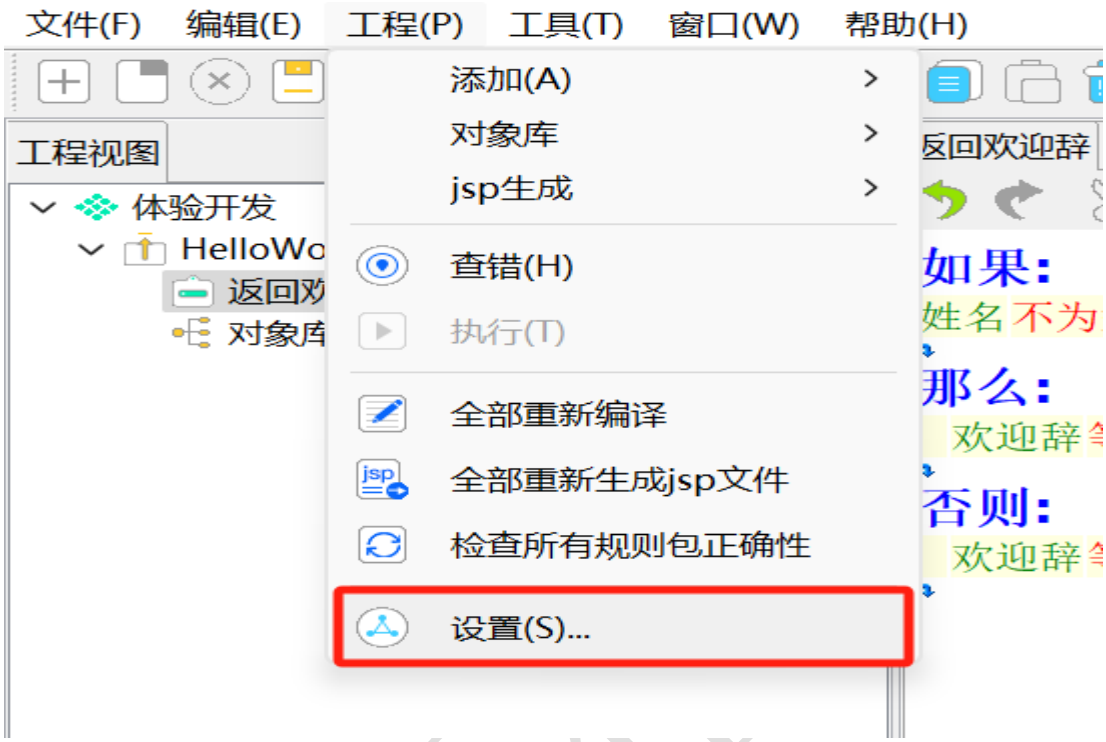
规则包发布时，需要根据该规则包生成对应的 java 代码，然后将该代码编译，生成.rsc 文件。将此文件输出到调用该规则包的应用程序的路径目录中，就完成了发布操作。

VisualRules 可以采用自动发布和手动发布两种方式，自动发布可以设置缺省的发布路

径，并且设置成在保存的时候，同时发布规则包。手动发布是在发布时，指定输出路径：

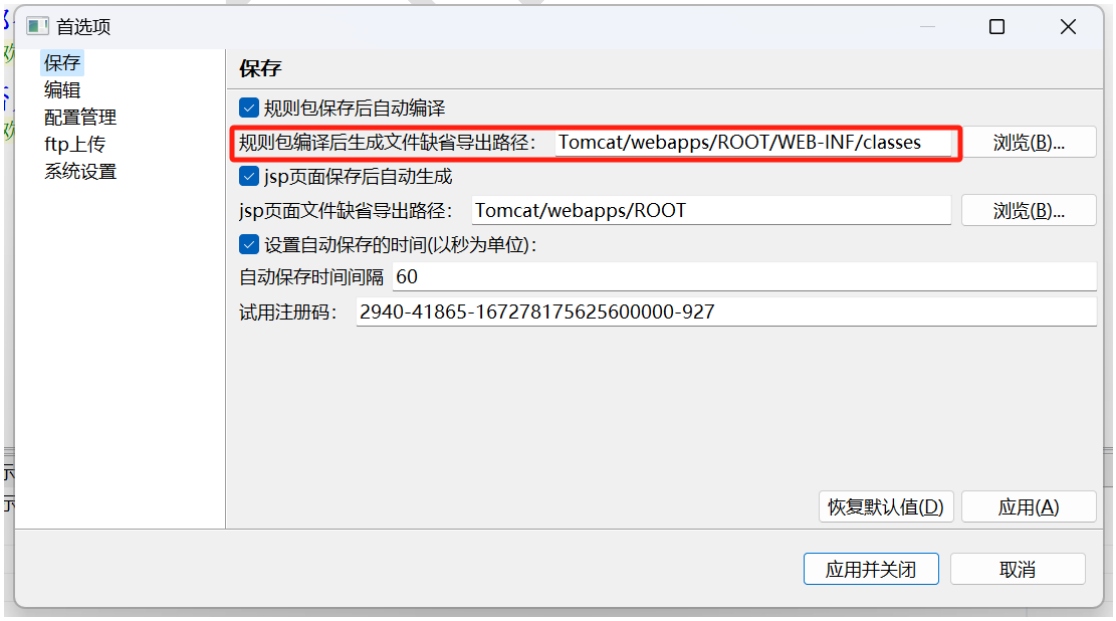
（1）自动发布设置

点击菜单栏中的“工程”，在弹出菜单列表中选择“设置”：



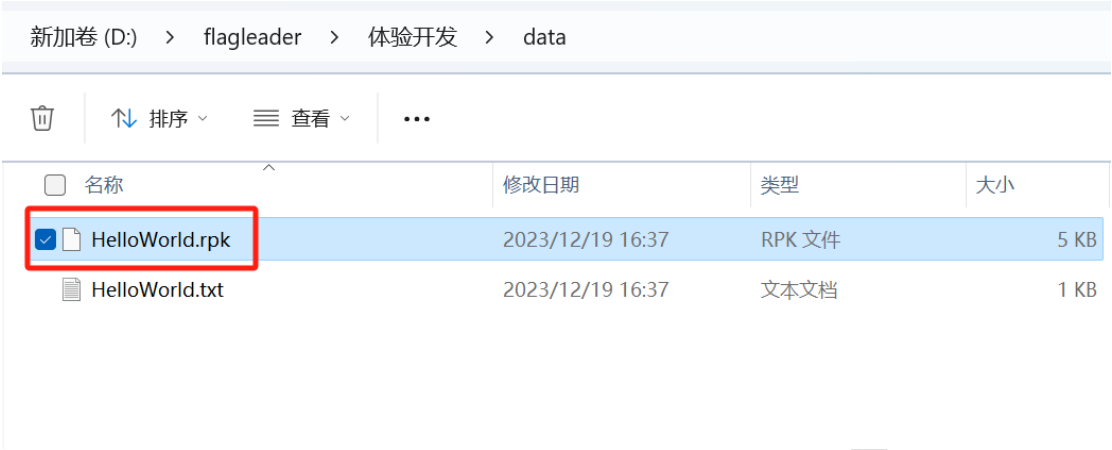
在弹出框中选中“规则包保存后自动编译选项”，并且设置缺省的输出路径。缺省情况下为“Tomcat\webapps\ROOT\WEB-INF\classes”，此路径是一个相对路径，相对于 VisualRules 的安装目录下。

因此如果 VisualRules 安装目录是 C:\visualRules，缺省情况下，点击保存后，会自动将规则包编译后的 rsc 文件保存到 C:\visualRules\ Tomcat\webapps\ROOT\WEB-INF\classes 中。



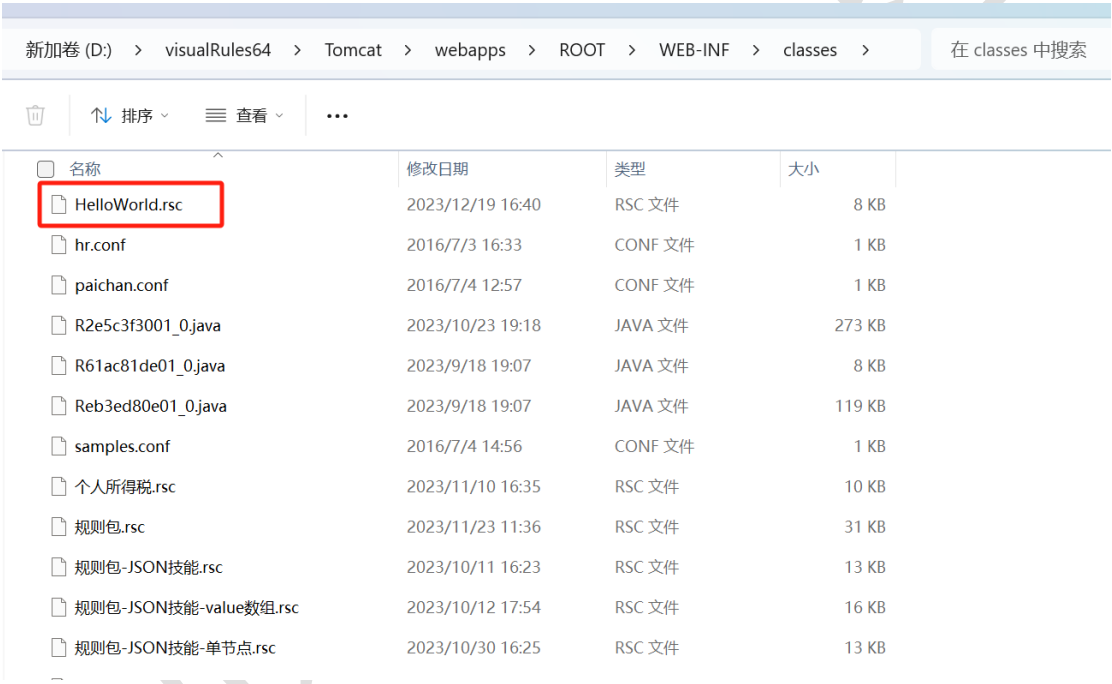
（2）自动发布

在规则配置器中，点击“ ”保存后，会将当前规则包以 rpk 文件的方式，将编译后的规则包以 rsc 文件的方式保存。查看规则工程目录的 data 下面的文件：



可以看到，在规则工程的 data 目录下，生成了 HelloWorld.rpk 的文件。其中 HelloWorld 就是规则包的可调用执行名。

查看规则配置器中设置的自动发布路径下，已经生成了 HelloWorld.rsc 文件：

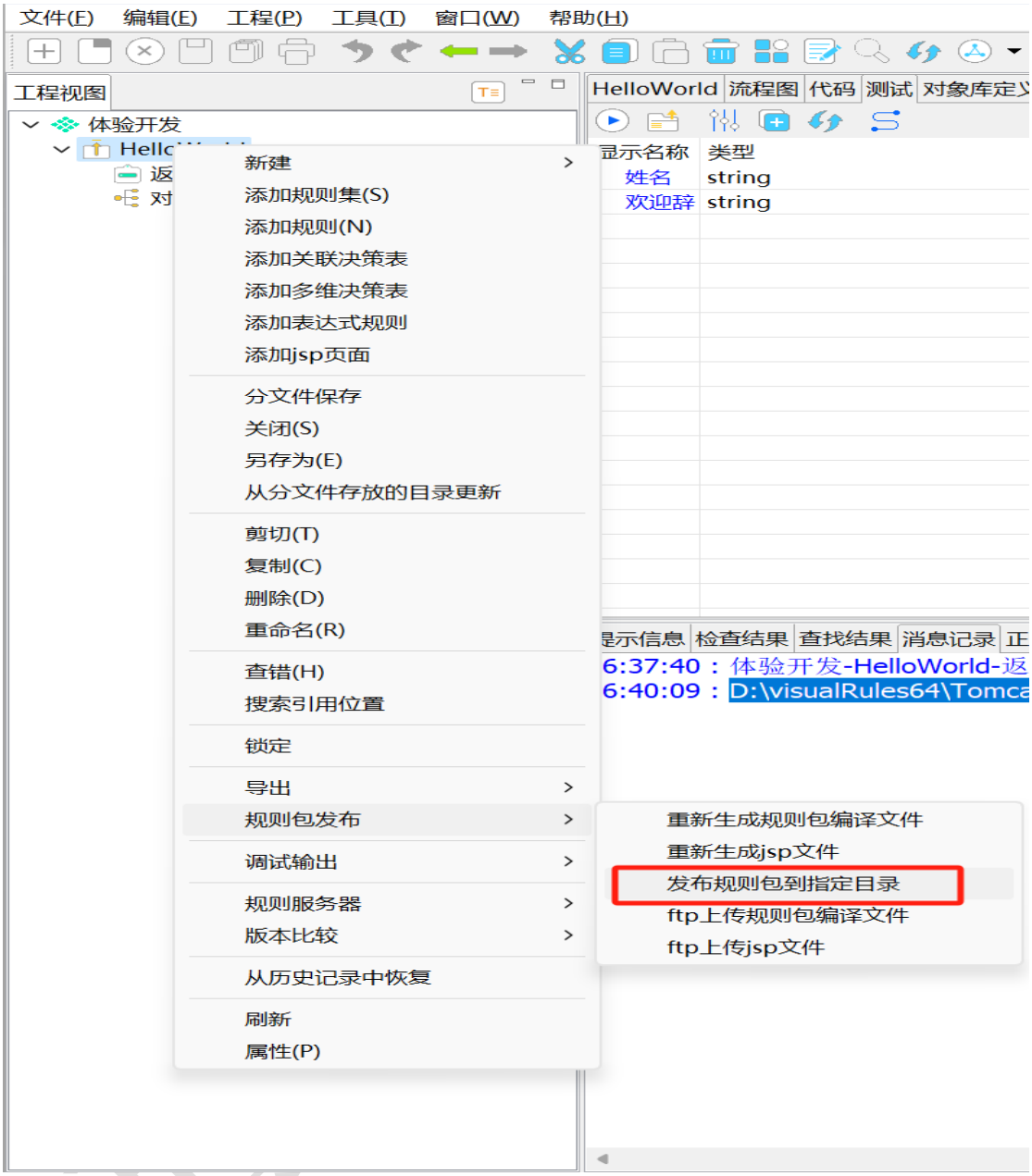


此目录是 VisualRules 自带的 Tomcat 的缺省工程路径，当规则包发布到此目录后，就可以通过 tomcat 的 http://localhost:8880/下面的 jsp 文件来访问这些规则包。

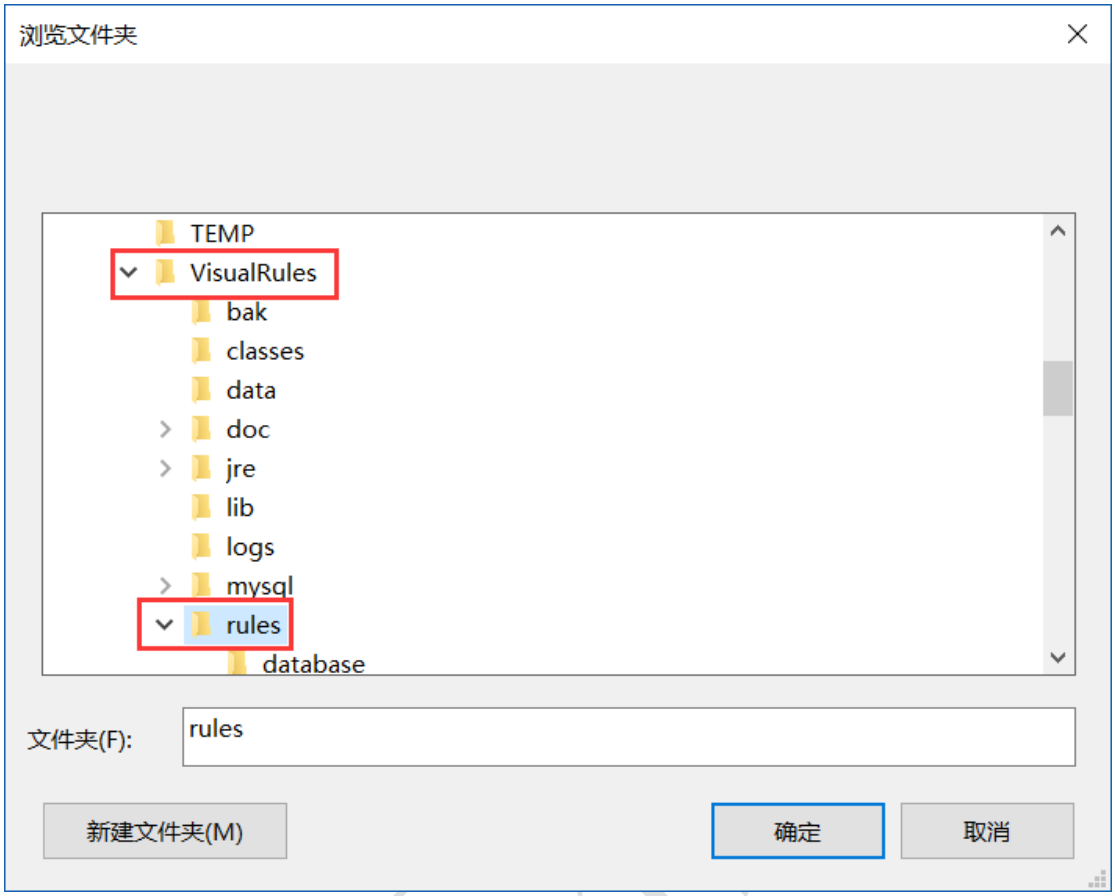
(3) 手动发布

为了能够通过规则服务来访问规则包，需要将规则包发布到 VisualRules 的规则服务路径中，此路径在 VisualRules 安装目录的 rules 目录下。

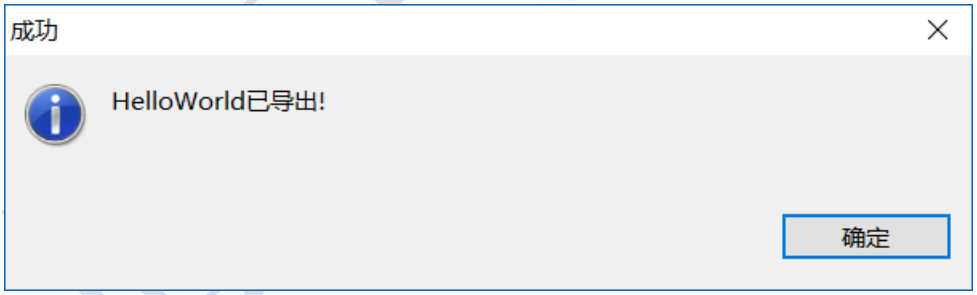
在规则包上点击右键，选择弹出菜单中的规则包发布→发布规则包到指定目录（也可选择“重新生成规则包编译文件”，它默认的路径为VisualRules自带的Tomcat的缺省工程路径）：



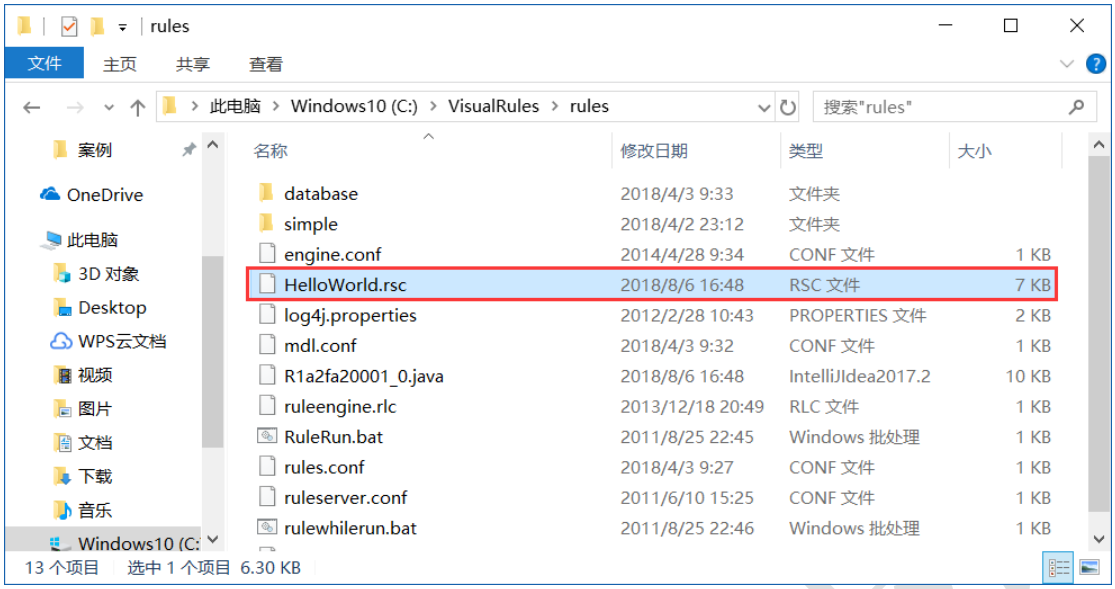
然后选择 VisualRules 安装目录的 rules 目录，点击确定：



出现以下弹出窗体后，表示发布成功：



然后查看 VisualRules 安装目录的 rules 目录下：

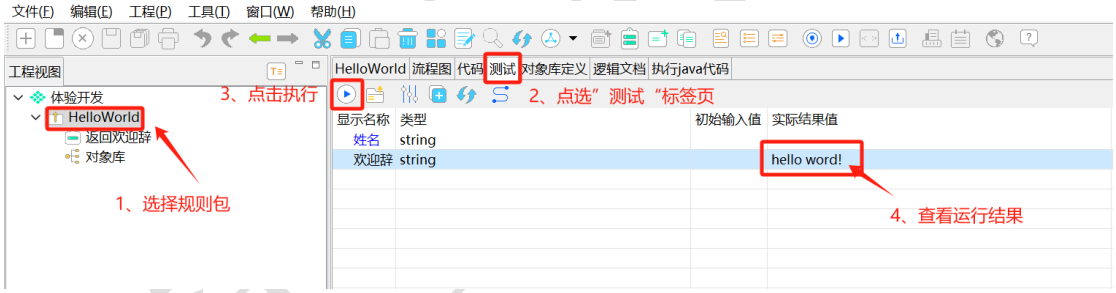


已经生成 HelloWorld.rsc 文件，说明发布成功。

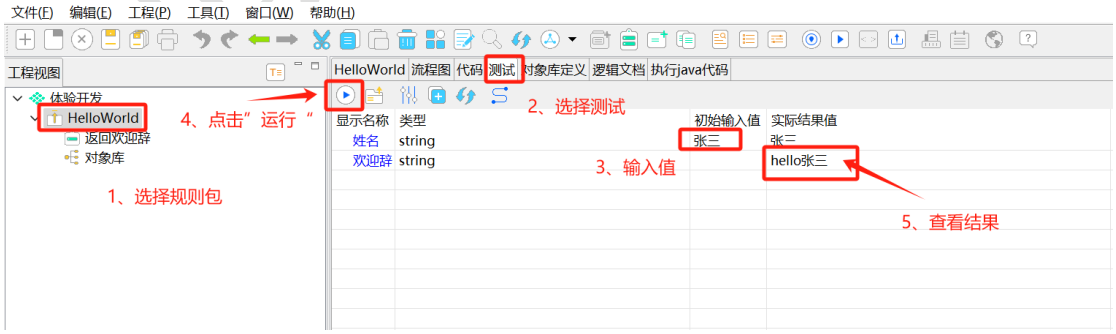
6. 测试规则包

可以直接在规则配置器中，测试规则包的执行情况。点击规则包后，在中间的编辑窗体的测试窗体中，可以输入传入值，点击执行后可以显示传出值：

(1) 姓名为空时的实际结果值：



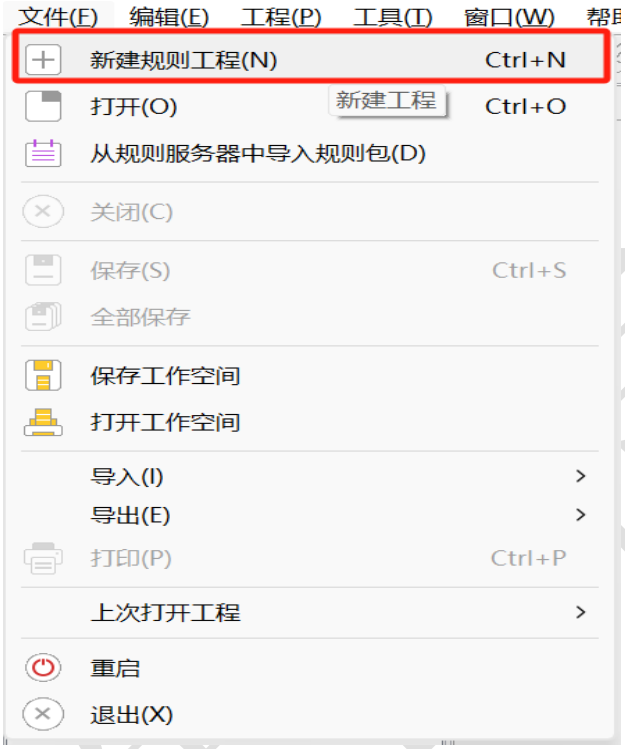
(2) 姓名不为空时的实际结果值：



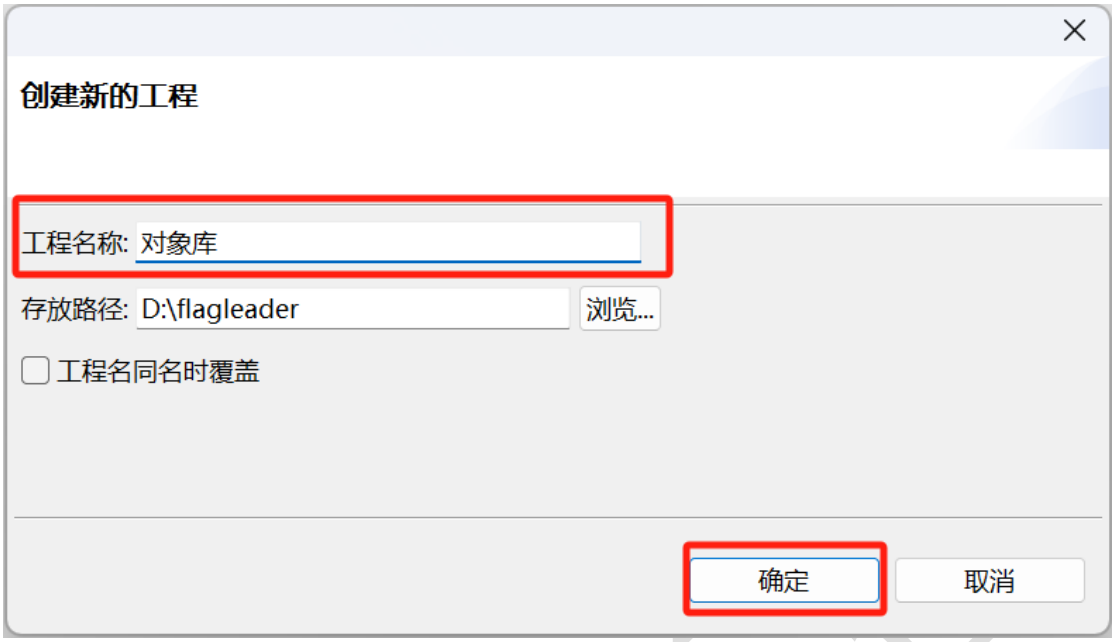
二、 对象库操作

2.1. 创建规则工程

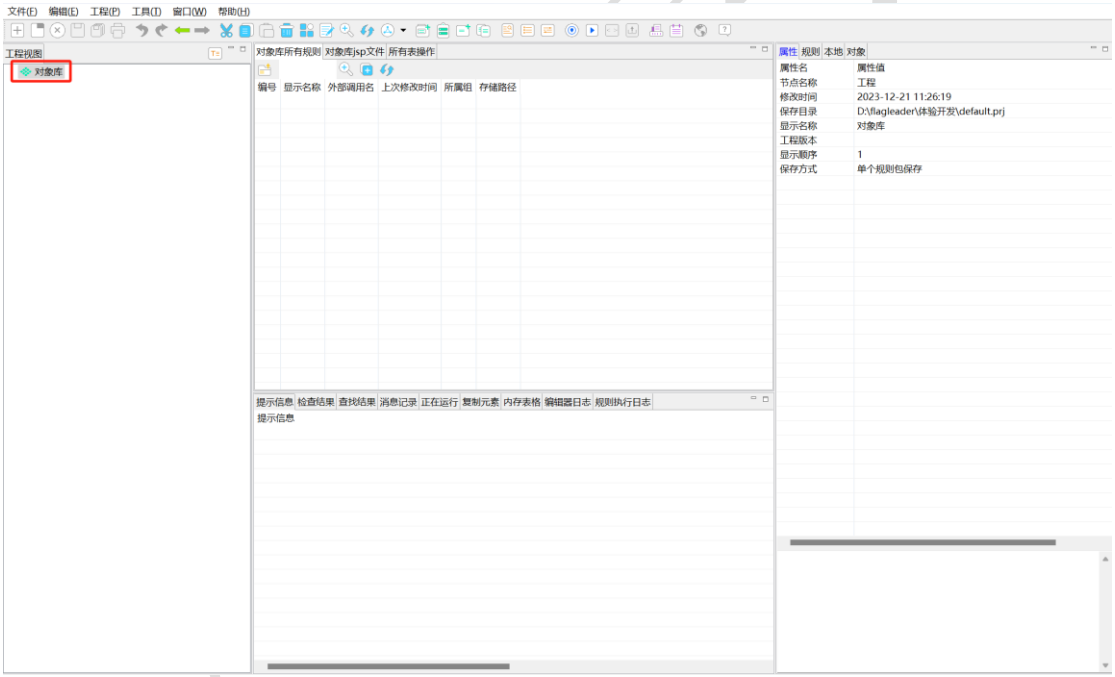
点击菜单栏上的“文件”，选择菜单项中的“新建规则工程”，如下图：



弹出对话框“创建新的工程”，在工程名称对应的文本框中输入“对象库”。如果要更改存放路径点击“浏览...”，选择自己的存放路径，如下图:



输入完成点击确定，如下图：



规则工程创建完成。

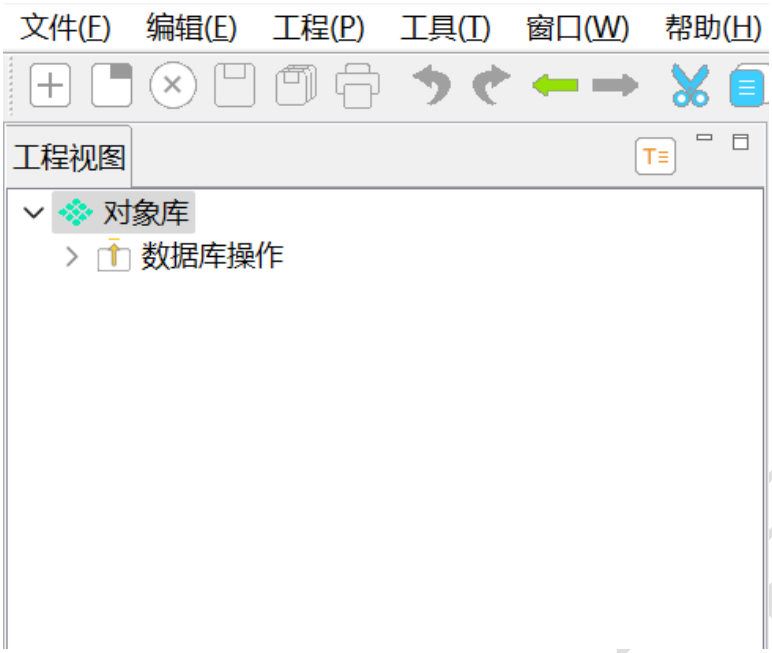
2.2. 对象库

2.2.1. 传入数据

1. 新建规则包

选中工程名为“对象库”的工程，右键点击“对象库”，选择菜单项中的“新建规则包”，如下图：

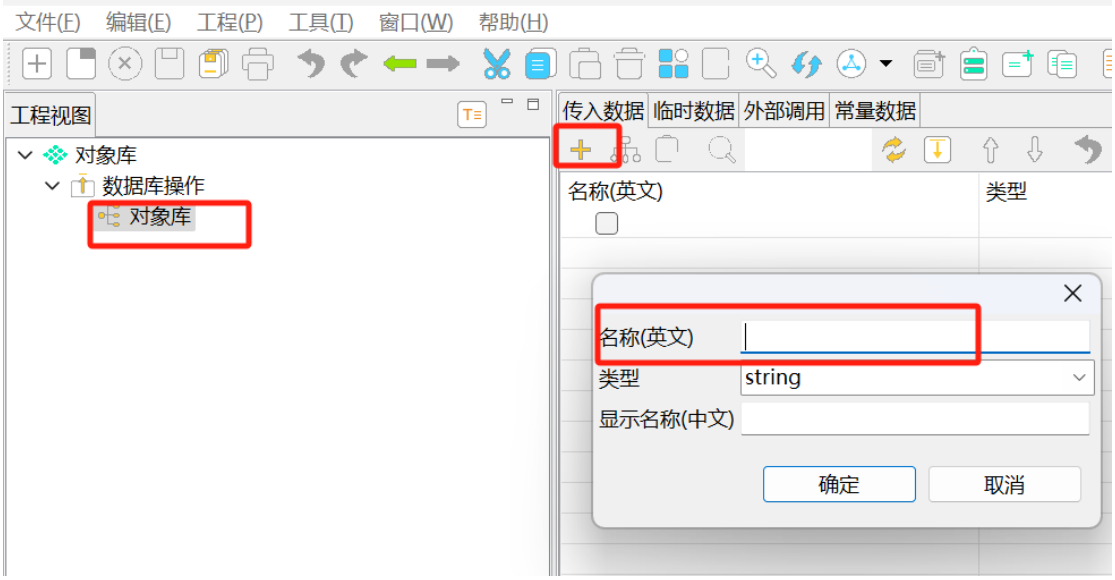
并将其命名为“数据库操作”，显示如下：



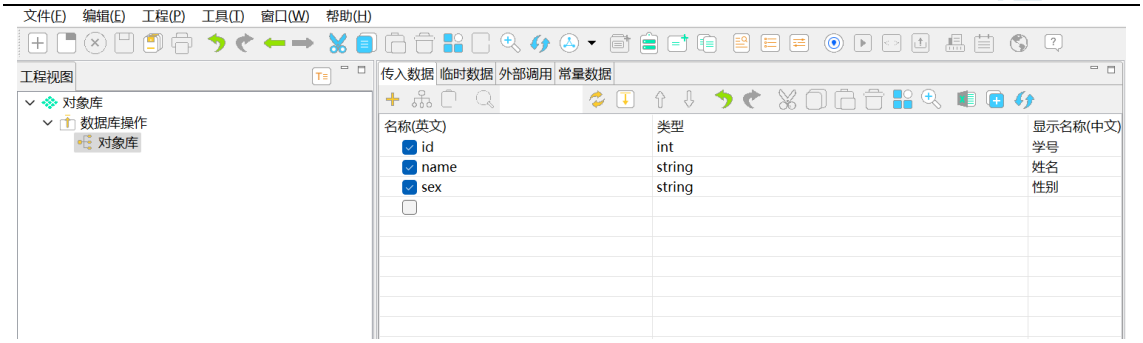
“数据库操作”规则包创建完成。

2. 创建对象库变量

打开“数据库操作”规则包，点击对象库，选择“传入数据”，点击“ ”添加对象库变量（也可直接在“名称”对应的列下双击输入），如图：

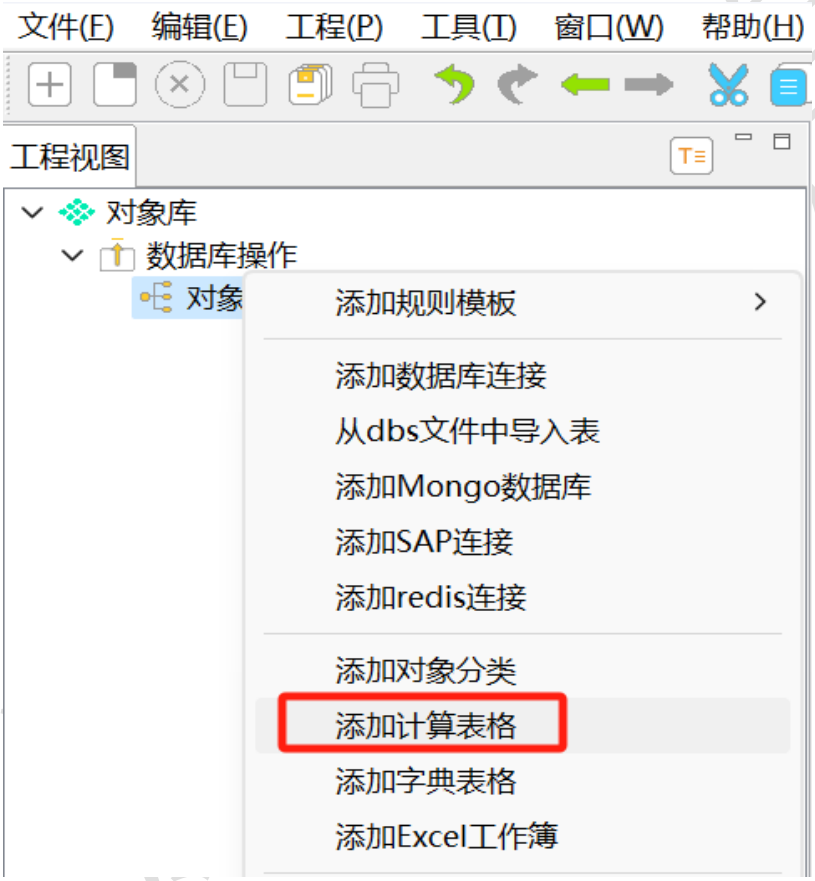


在弹出的对话框里依次填入名称、类型、显示名称。（名称必须为英文，类型默认为 String，这里根据自己的需求去在下拉框中选择，显示名称则可以为中文。）以“学号（id）、姓名（name）、性别（sex）”为例：

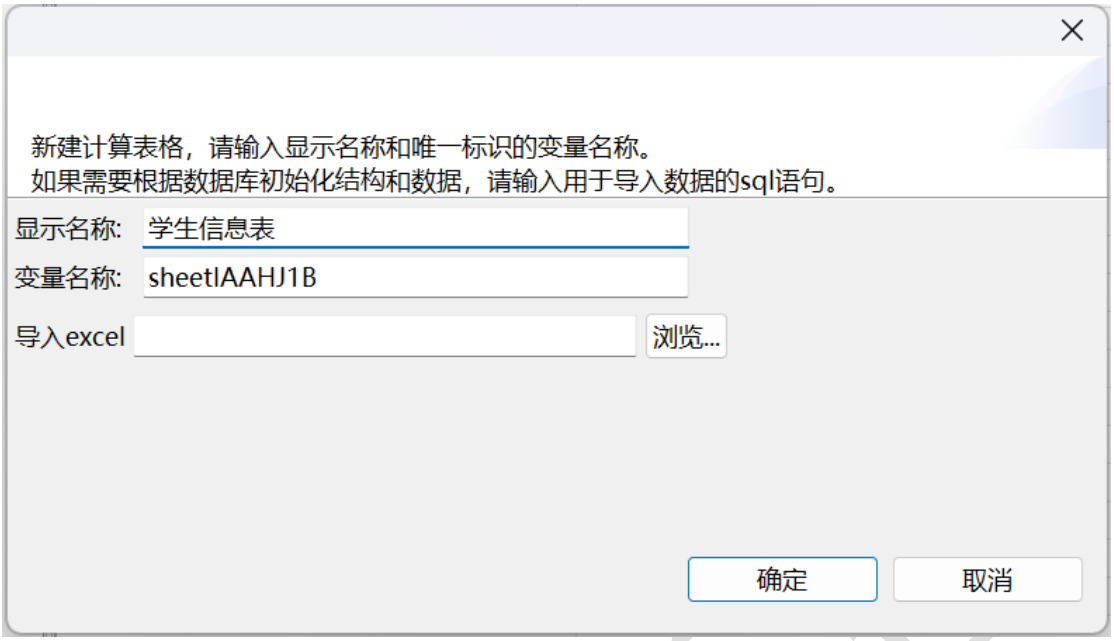


3. 添加计算表格

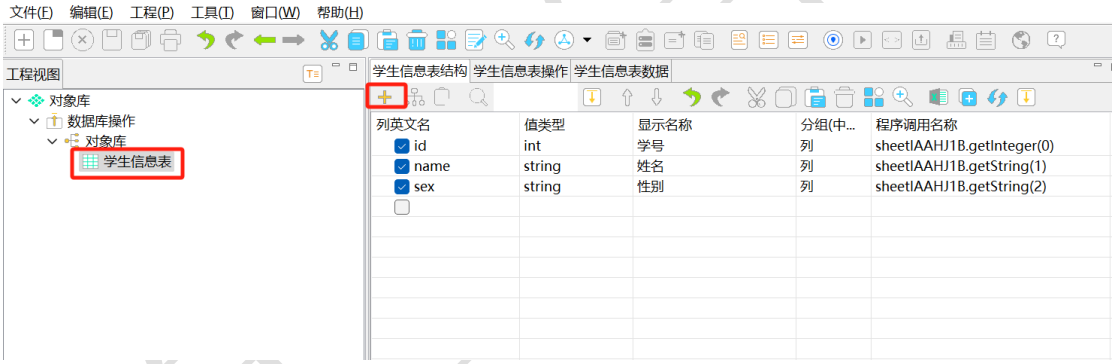
对象库添加完成之后我们需要新建一个“计算表格”，用以储存计算结果值。点击“对象库”，右键选择“添加计算表格”，如下图：



并将其命名为“学生信息表”：

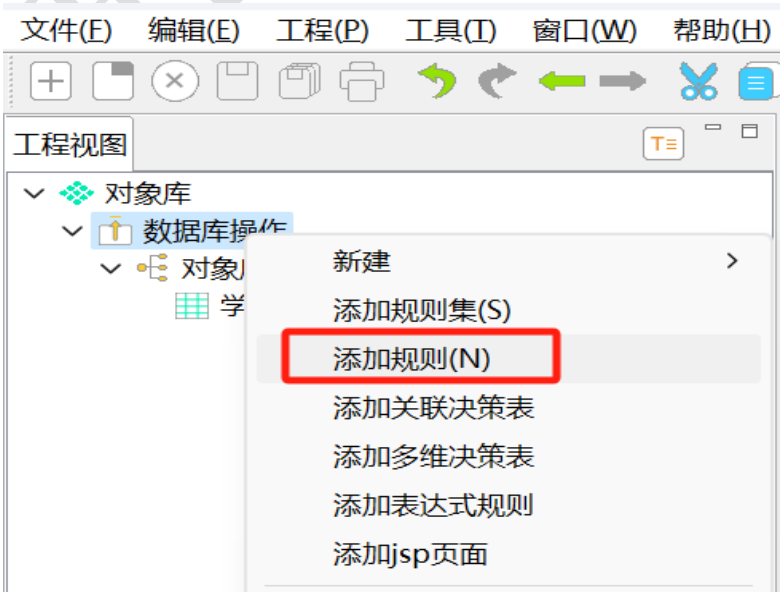


选择“学生信息表”点击将上面“对象库”“传入数据”中的变量逐个添加到“计算表格”（也可直接从“对象库”复制到“计算表格”）：

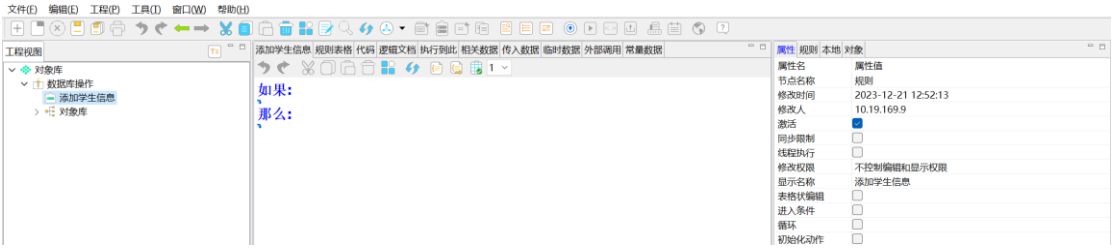


4. 编写规则

右键点击“数据库操作”规则包，选择添加规则，并将其命名为“添加学生信息”：



打开“添加学生信息”规则：



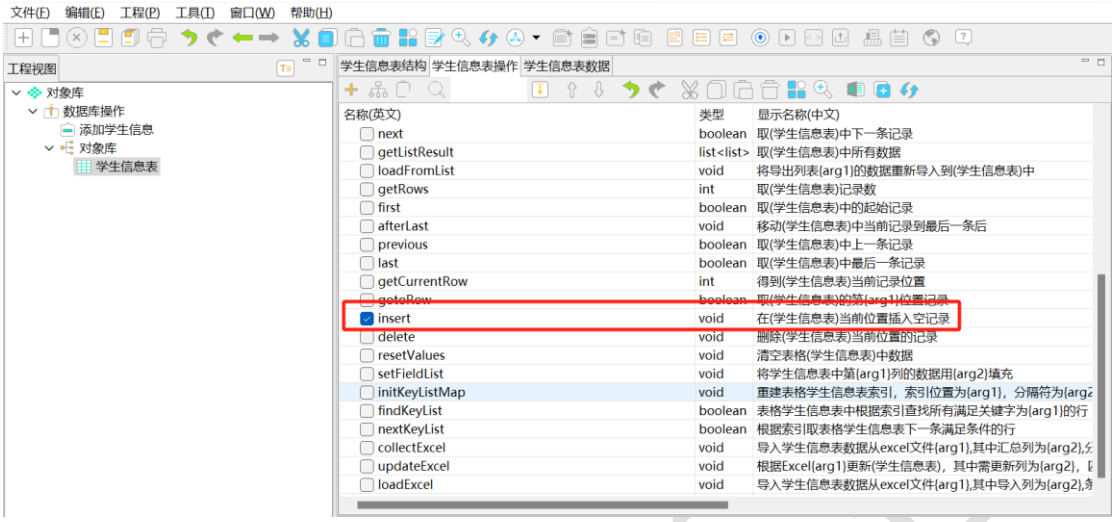
在右侧属性栏中找到“其他表格”，并将其属性值更改为“允许设置其他表格的列”。

属性名	属性值
节点名称	规则
修改时间	2023-12-21 12:52:13
修改人	10.19.169.9
激活	☑
同步限制	☐
线程执行	☐
修改权限	不控制编辑和显示权限
显示名称	添加学生信息
表格状编辑	☐
进入条件	☐
循环	☐
初始化动作	☐
否则动作	☐
异常动作	☐
遍历表格	☐
执行顺序	1
否则如果	双击选择设置
异常处理	抛出新的异常
提示信息	
其他表格	不允许设置其他表格的列
执行轨迹	记录
对象库快照	不进行快照
匹配后处理	不处理
流程图显示	显示规则定义内容
独立编辑	不支持
生效日期	
失效日期	

与 Java 插入数据一样，规则配置插入数据时也需要 insert，这里对该方法进行了封装，

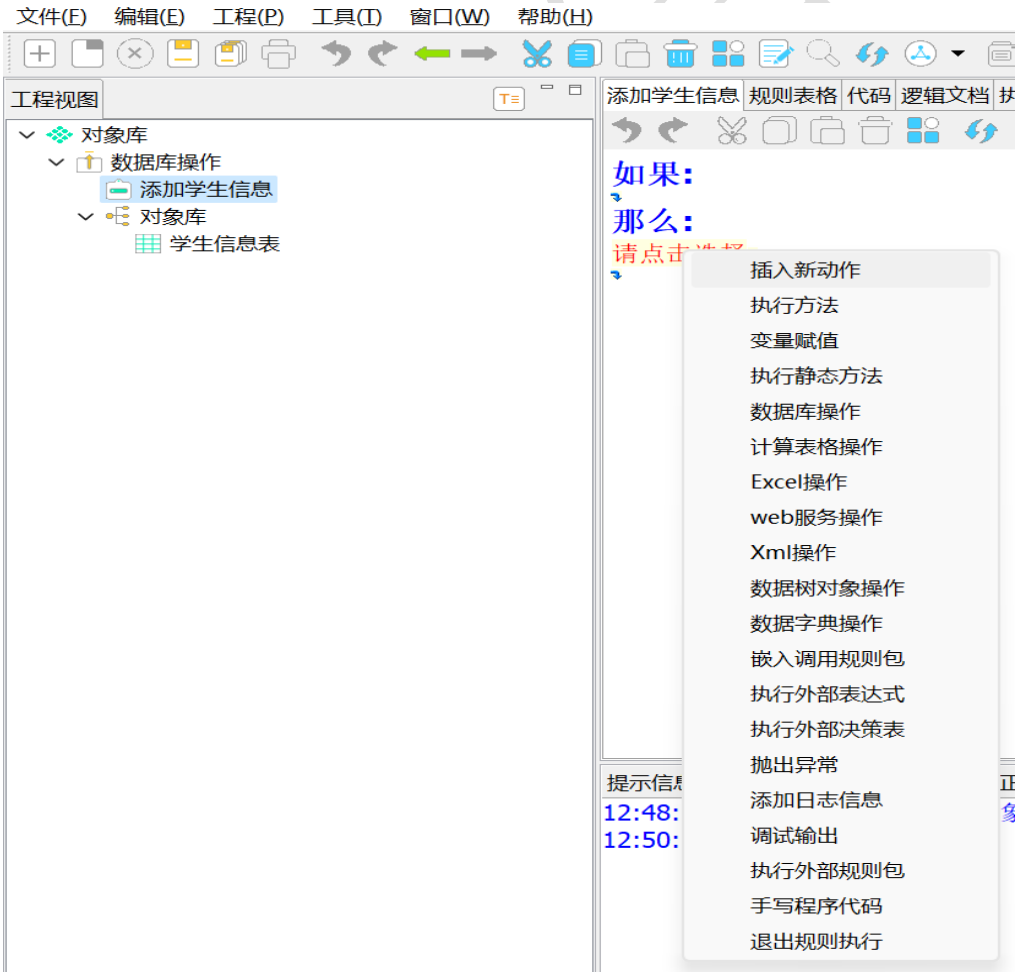
我们直接选择“在(arg)当前位置插入空记录”

点击“学生信息表”，选择“学生信息表操作”，在里面找到“名称”为“insert”的方法并勾选。



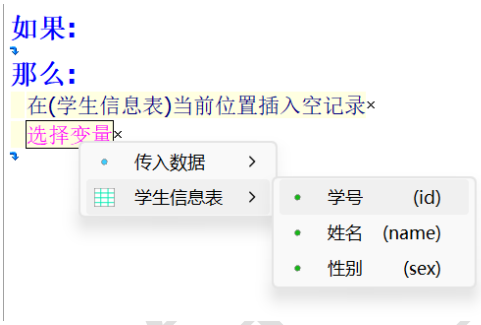
（注：由于 World 中图片是等比例缩小，因此如有不清晰可单击图片选择“🔍”查看原图。）

点击中间输入框“那么”下的“🔽”，出现“请点击选择×”，点击出现如下选项：

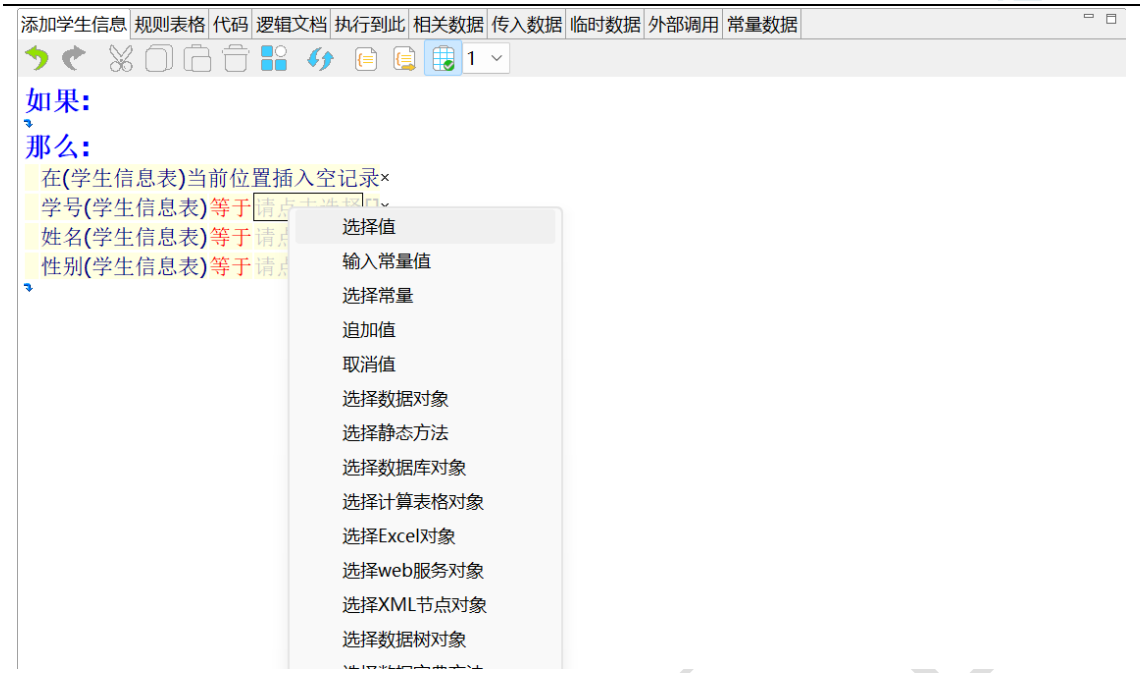


选择“执行方法”出现“选择方法×”，选择“结果集”下面的“在(学生信息表)当前位置插入空记录”

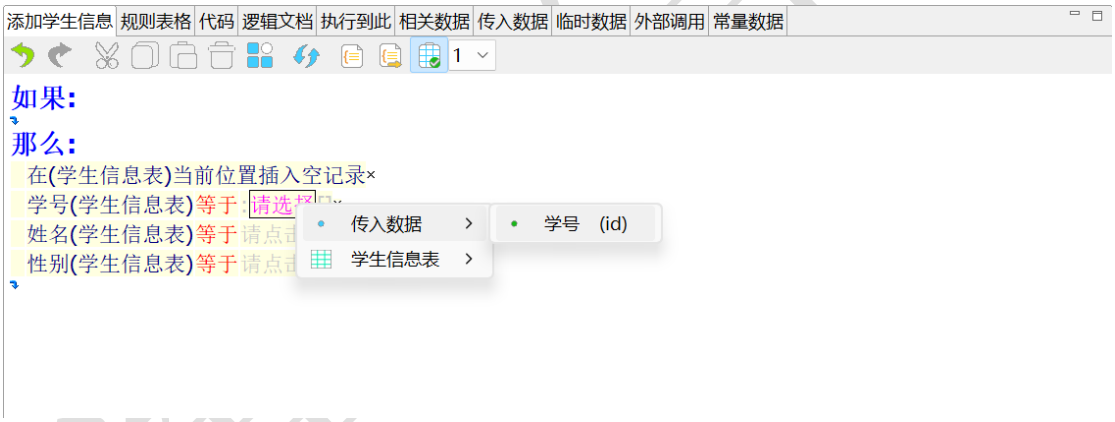
同上，选择“变量赋值”出现“选择变量×”，点击依次选择“学生信息表”中的“学号”、“姓名”、“性别”：



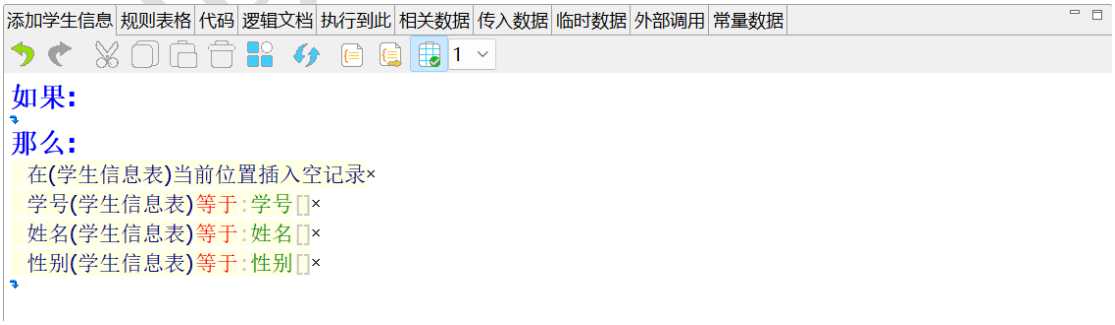
依次给上面添加的“学生信息表”中的变量进行赋。



选择“传入信息”。（因为“学生信息表”里的“学号（id）”设置的为 int 类型，所以后面“传入数据”中所匹配到的选项只有一个为 int 类型的“学号（id）”）

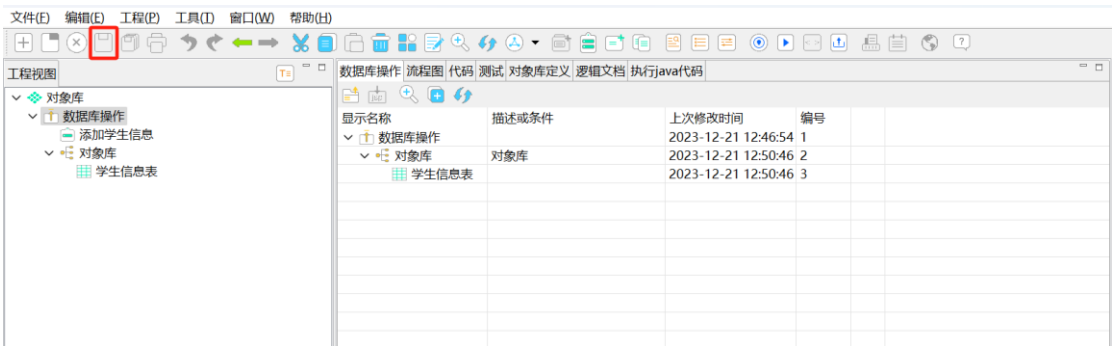


根据“学号（id）”的操作，将另外的两个变量也进行对应的赋值。

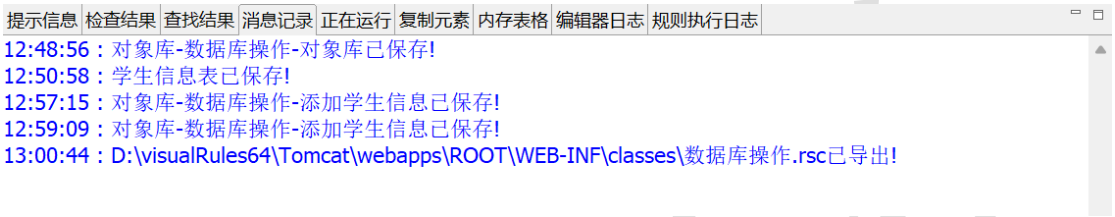


5. 保存并编译

规则逻辑完成后，我们需要对规则进行保存、编译，步骤如下图所示：



点击“全部保存”按钮，然后在消息窗口会有如下图所示的记录：



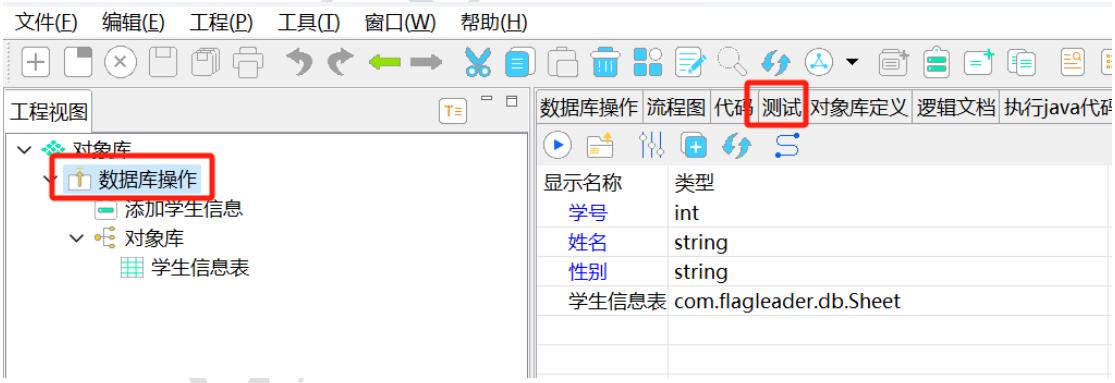
这两句消息记录表示该规则包已成功保存并编译。若该规则包出现错误，在消息窗口将会出现红色字体的提示。

6. 测试

(1) 单个测试

完成以上操作之后，一个简单的规则项目就创建完成了，下面进行测试。

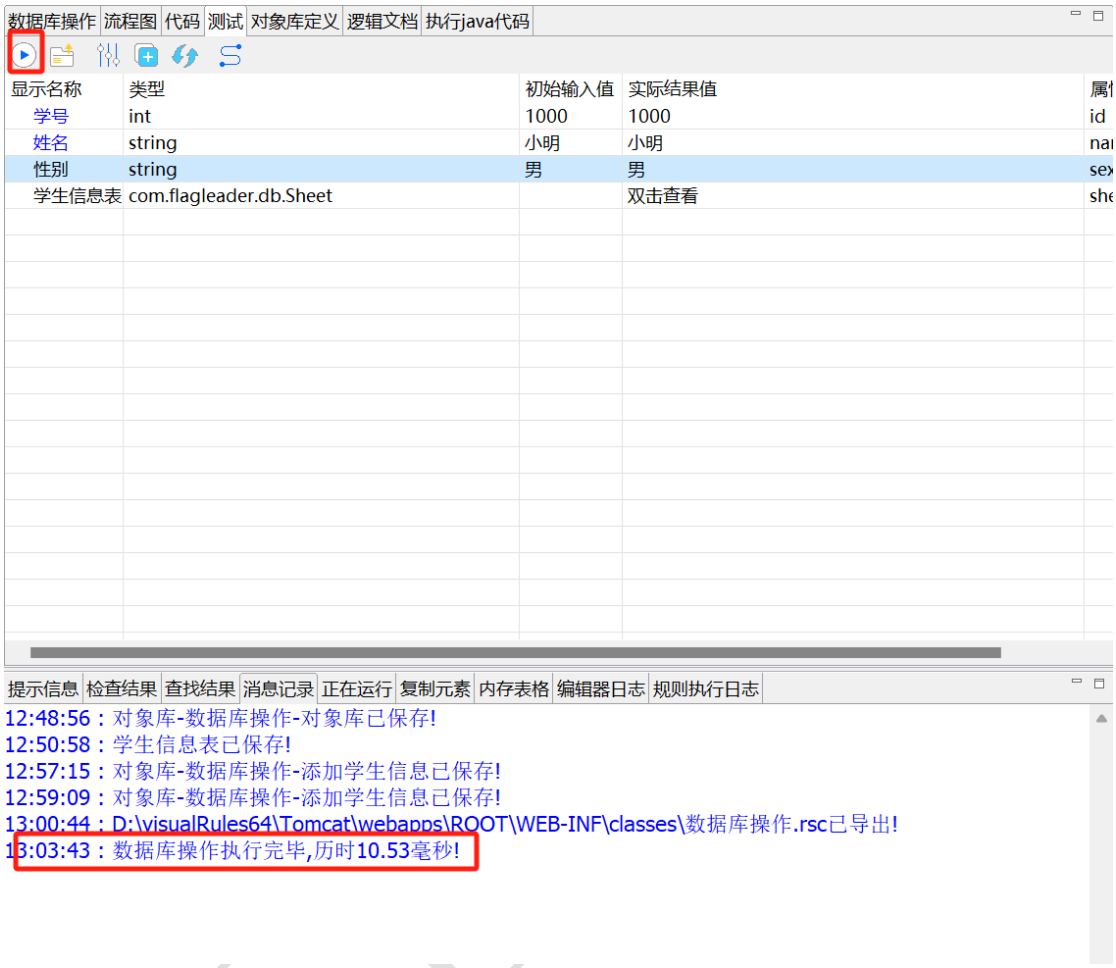
点击“数据库操作”规则包，在右侧所出现的页面中点击“测试”：



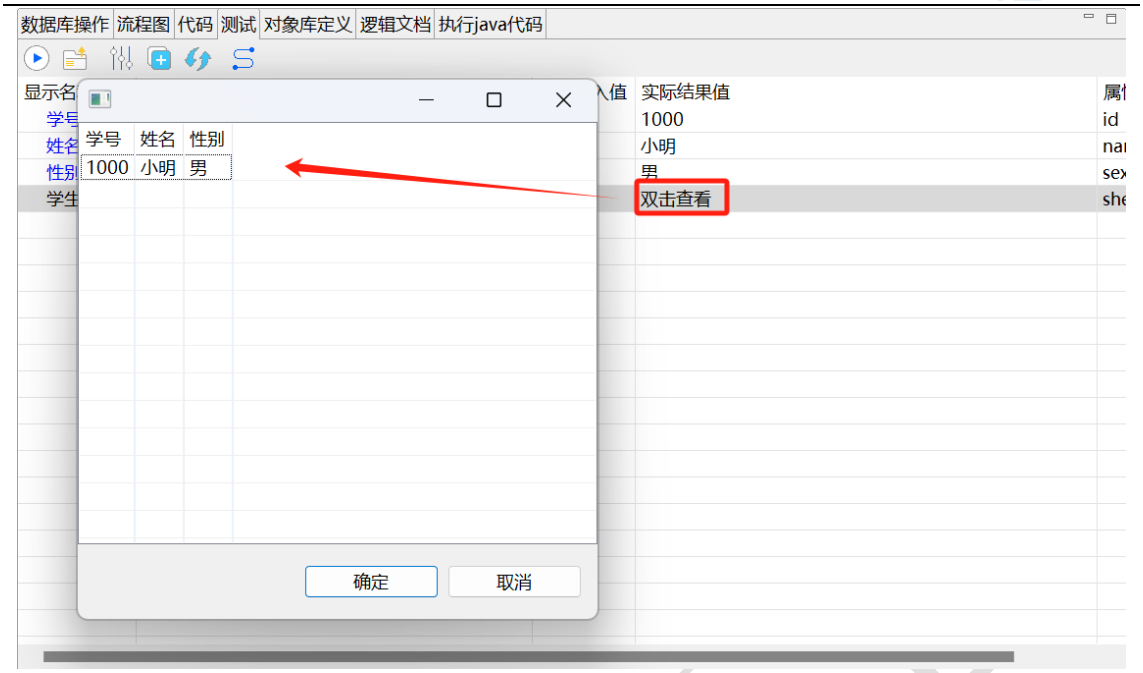
测试界面中，每个字段对应的有“初始输入值”和“实际结果值”，其中“初始输入值”为我们测试的时候所输入的测试值，而“实际结果值”则是根据我们输入的“初始输入值”在规则中进行计算所得到的结果。

双击“初始输入值”列下的空格，依次对每个变量进行赋值：

输入完成之后，点击左上角的“ ”执行图标进行测试，执行完成“实际结果值”这一列下面会出现测试的计算结果数据，以及在下方消息窗口“消息记录”中出现执行过程所经历的时间：



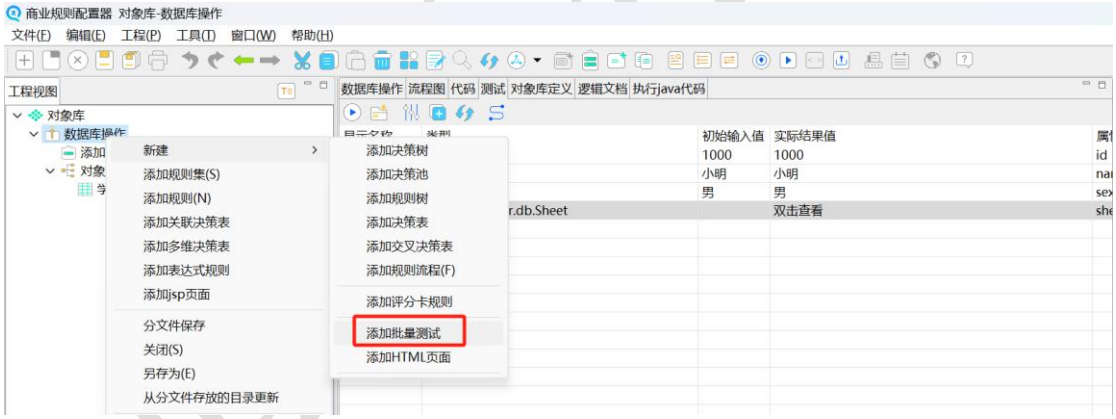
点击“实际结果值”列、“学生信息表”行对应的“双击查看”，弹出框中数据正常则说明程序正确（也可在“对象库”下面的“学生信息表”中点击第三列“学生信息表数据”进行查看）：



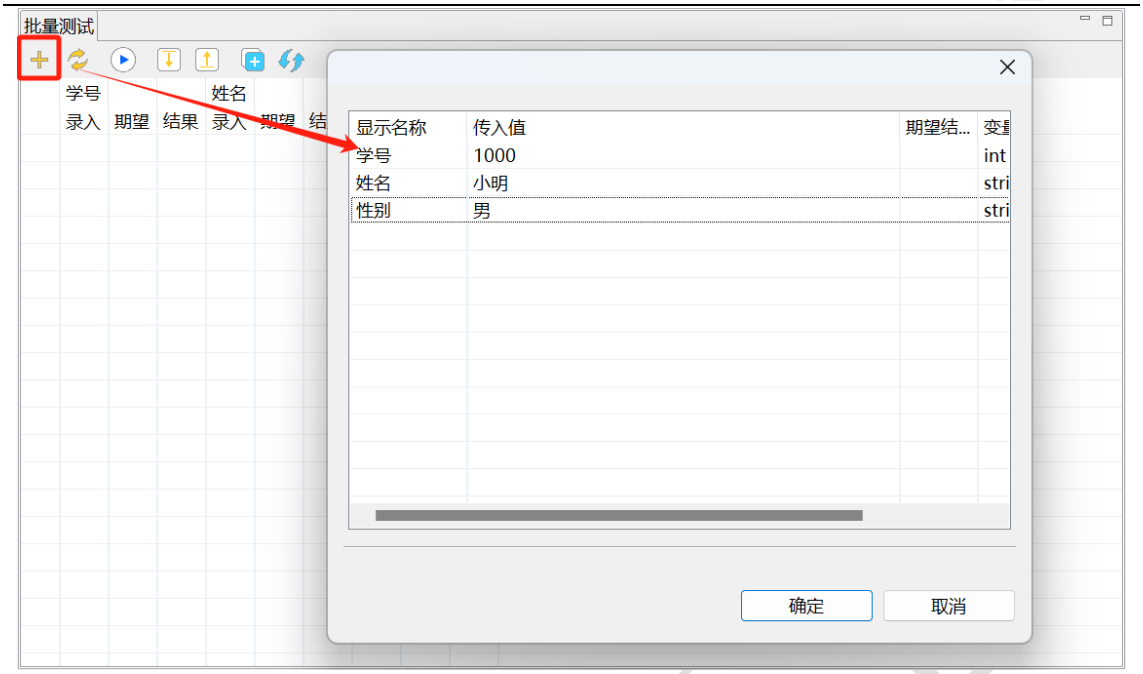
(2) 批量测试

在实际运用中往往会有多条数据需要进行测试，而通过上面的方法逐个进行测试，效率就比较低了，这是我们可以新建一个“批量测试”来同时进行测试多条数据。

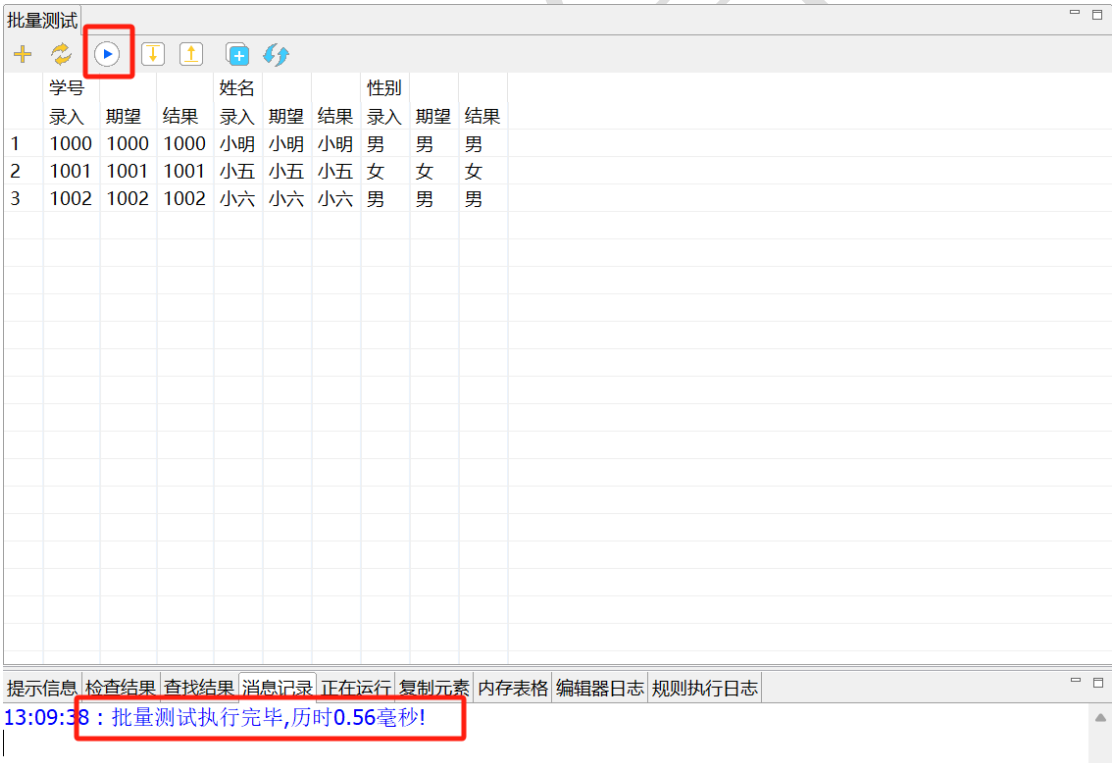
右键“数据库操作”选择“新建”下的“添加批量测试”：



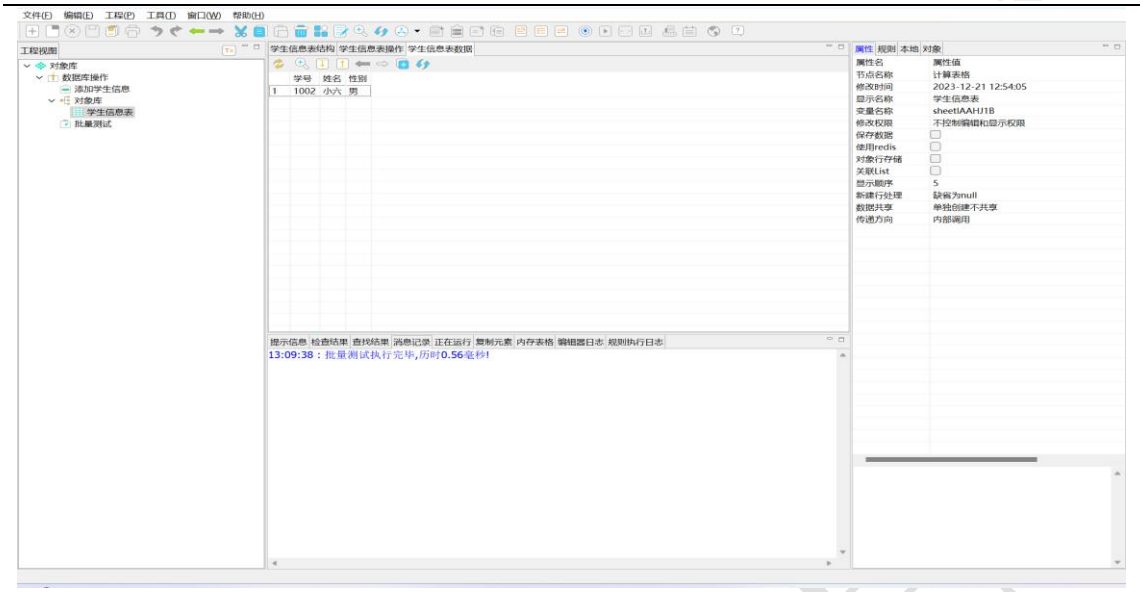
在“批量测试”界面中，点击右上角的“ ”添加“传入值”和“期望结果值”，完成后确定，再次点击右上角的“ ”添加下一条测试数据：



所有数据都添加完成之后，点击“ ”执行，跟上面的测试一样，批量测试也会有“结果值”以及“消息记录”中的执行所用时长：



打开“对象库”下的“学生信息表”可以查看表中测试所得到的结果值。（由于本案例是直接对“对象库”中的“传入数据”进行赋值，因此没有遍历表格（类似于 Java 中的 list、map 循环赋值），所以这里表中的数据只有最后一条的记录。）



以上就是通过规则配置器完成的一个简单的规则项目，它是由“对象库”中的“传入数据”赋值，到规则中进行传值，再将结果值保存到计算表格。相比用代码写整个流程，规则配置器则只需要简单的几步配置、赋值就能达到所要的效果，执行效率大大的提高。

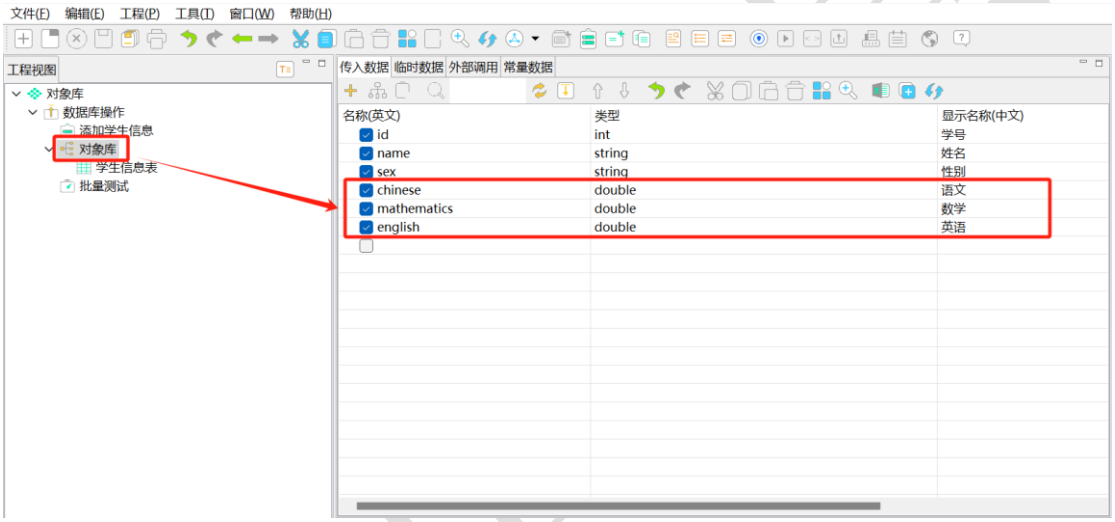
2.2.2. 临时数据

在 Java 中为了程序的有效执行，我们往往会添加一些 int i=0;String str=null;的变量，而这些新定义的变量在最终的结果中通常是用不到的。因此，在规则配置器中我们把这样的变量归纳进“临时数据”，统一定义在“对象库”“临时数据”中。

例如，要计算学生考试的总成绩以及每门课程的平均分，而这两个值是通过计算所得出的，原本的“传入数据”中是没有的，所以“总分”和“平均分”我们可以把它定义在“临时数据”中。如下：

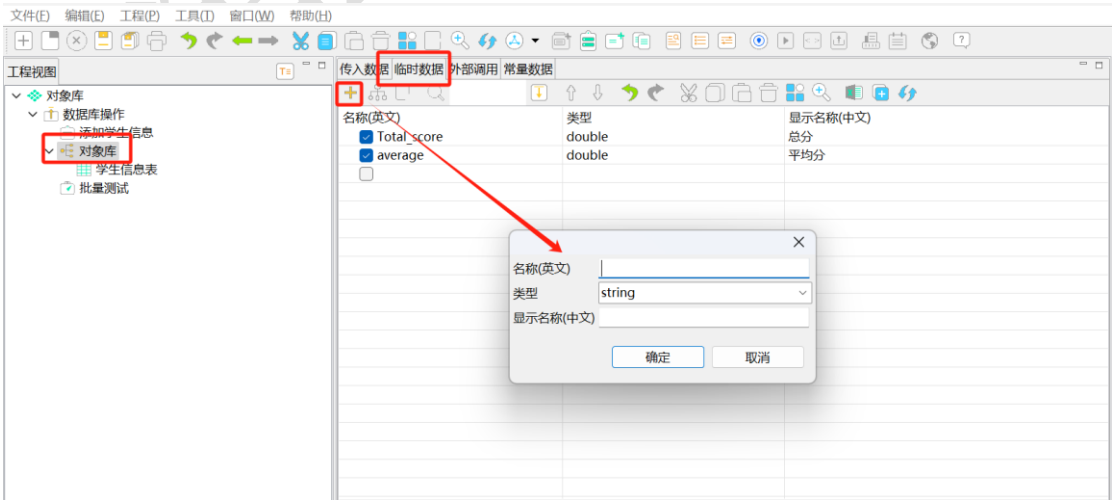
1. 添加传入数据

在“对象库操作”规则包下，“对象库”中“传入数据”添加如下字段“语文（chinese）、数学（mathematics）、英语（english）”：



2. 添加临时数据

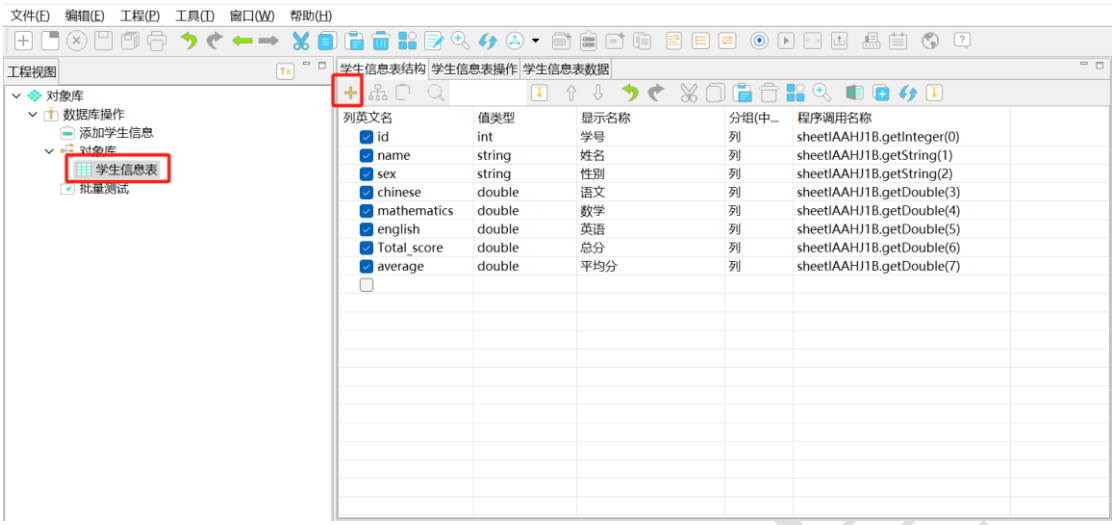
依次点击“对象库”“临时数据”中左上角的“ ”添加“总分（Total_score）、平均分（average ）”字段：



3. 添加计算表格数据

在“学生信息表”中依次添加“语文（chinese）、数学（mathematics）、英语（english）、

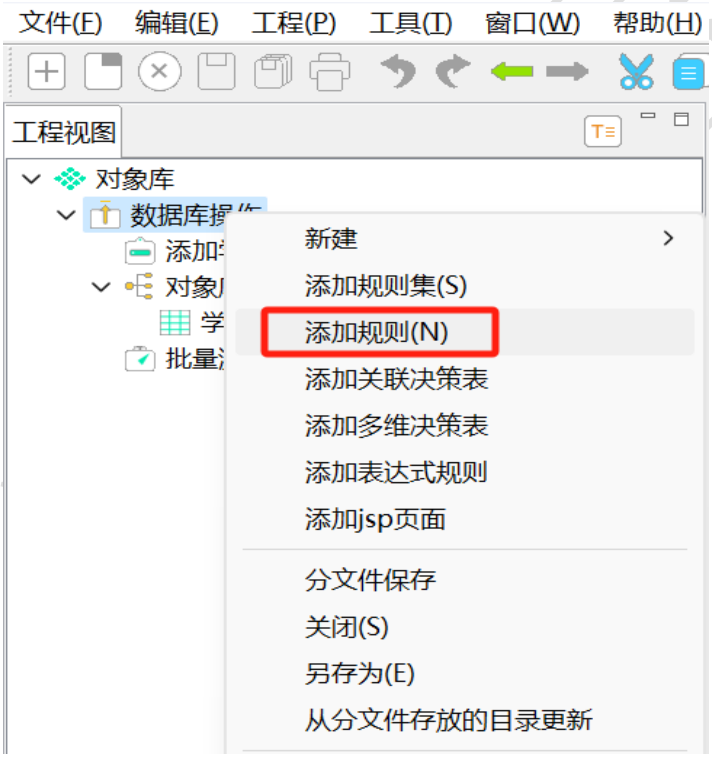
总分（Total_score）、平均分（average ）”：



4. 添加规则

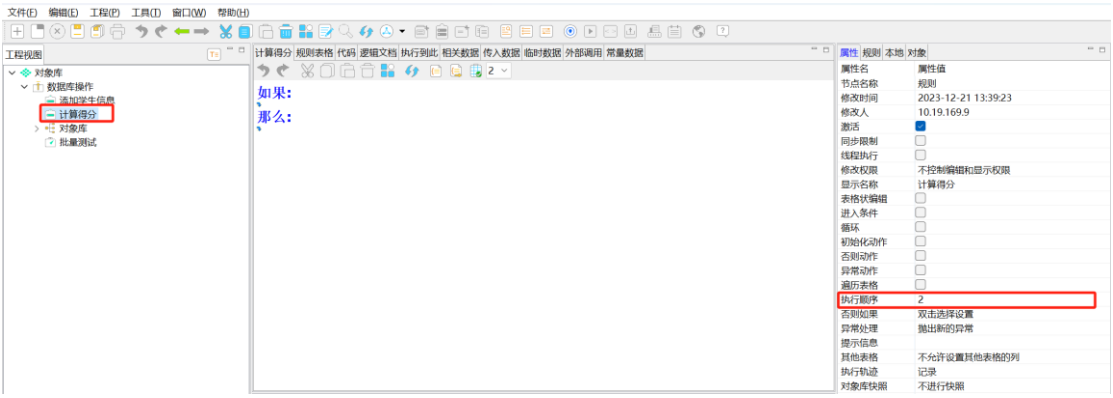
(1) 计算得分

右键“数据库操作”选择“添加规则”，并将其命名为“计算得分”：

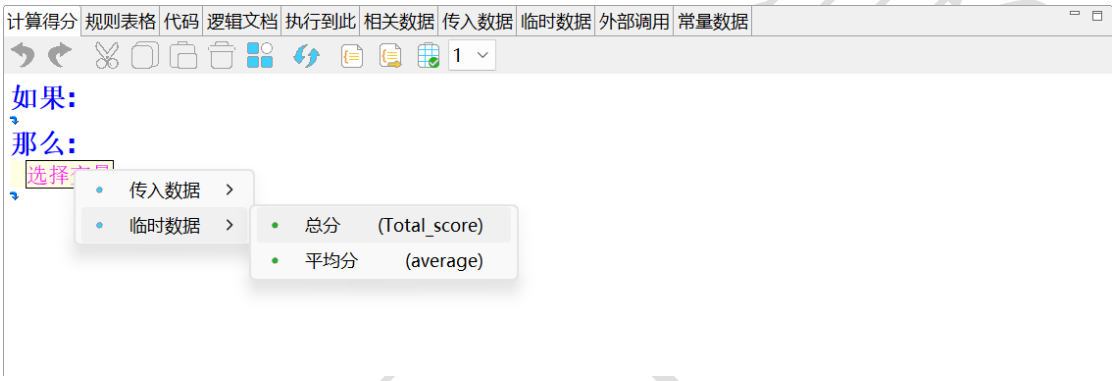


由于规则的执行顺序是从上往下依次执行的，而在这里我们要将计算结果“总分”、“平均分”存入计算表格，因此，要将“计算得分”这条规则置于“添加学生信息”规则的前面。

点击“计算得分”规则，在右侧属性栏中将属性名为“执行顺序”的属性值所在行双击，出现下拉框，将属性值改为“1”，点击空白处完成顺序变更。

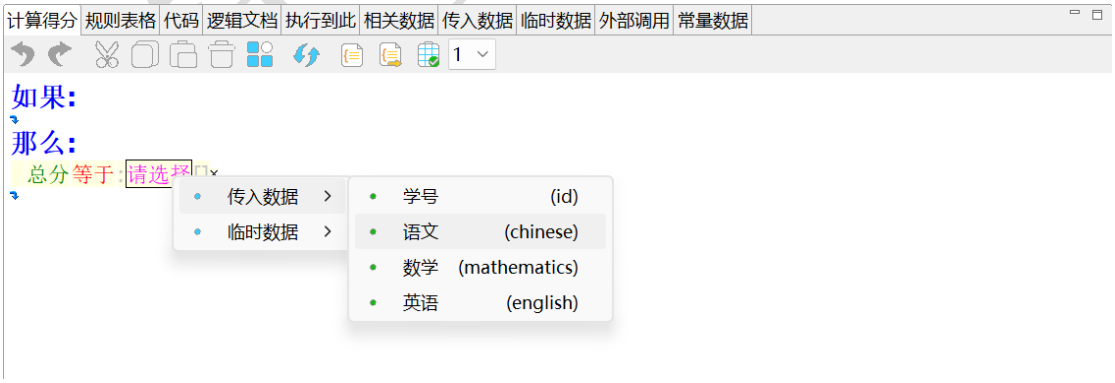


点击中间输入框“那么”下的“ ”，出现“ 请点击选择× ”，点击选择“变量赋值”，选择“临时数据”中的“总分”：

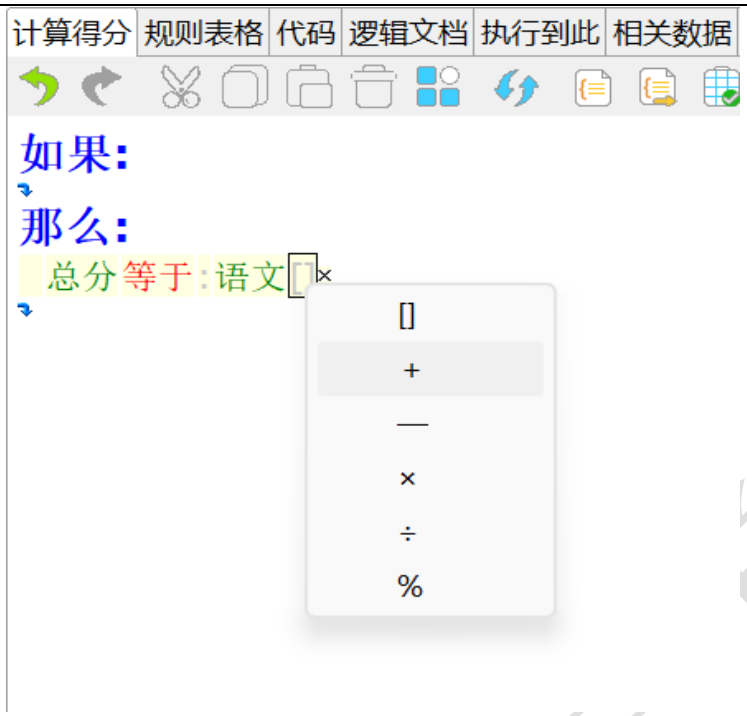


在规则引擎中，想要计算出一个值，只需通过简单的数学计算就可得出结果（总分 = 语文 + 数学 + 英语）、（平均分 = 总分 ÷ 3）

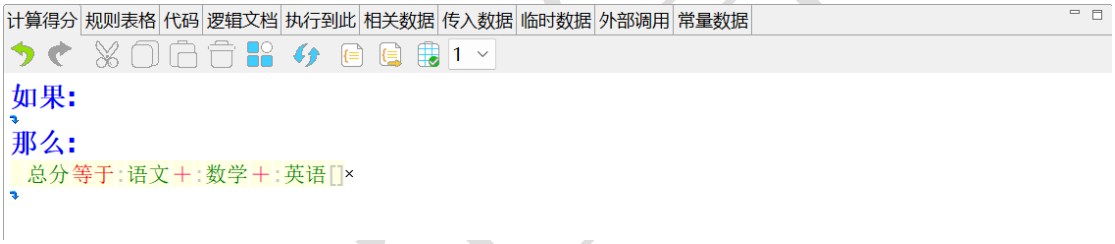
点击“请点击选择[]×”选择“选择值”，点击“请选择[]×”选择“传入数据”下的“语文”：



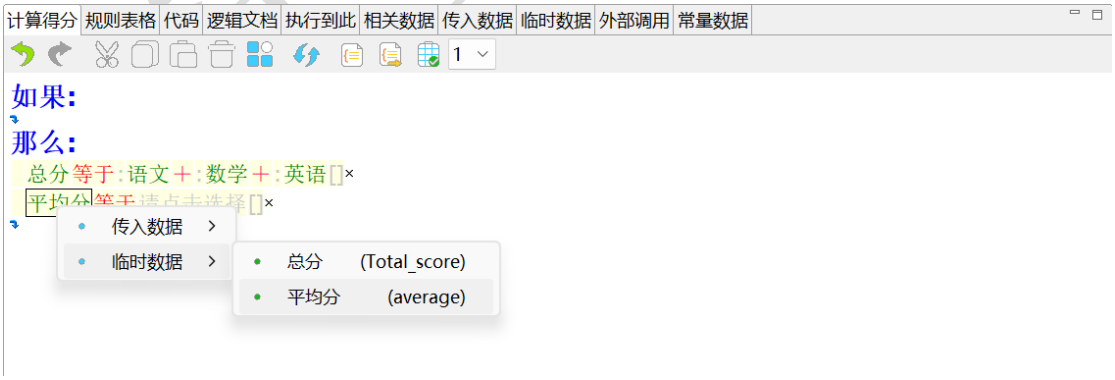
点击刚刚添加的“语文”后面的“[]”，选择你需要的“数值间操作符”，这里我们是要求和，因此选择“+”：



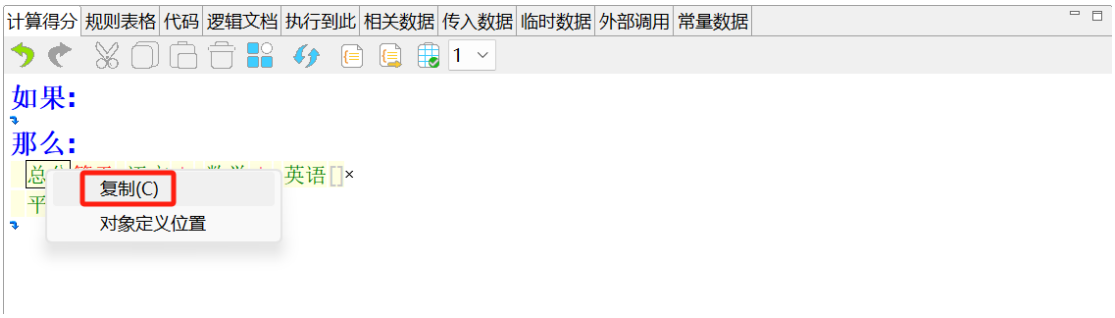
同上，添加“数学”和“英语”：



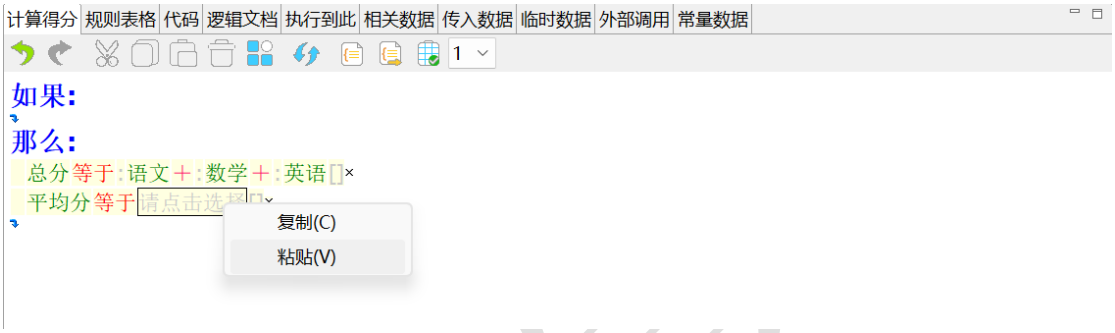
算出“总分”下面我们可以来求“平均分”。点击中间输入框“那么”下的“ ”，出现“请点击选择×”，点击选择“变量赋值”，选择“临时数据”中的“平均分”：



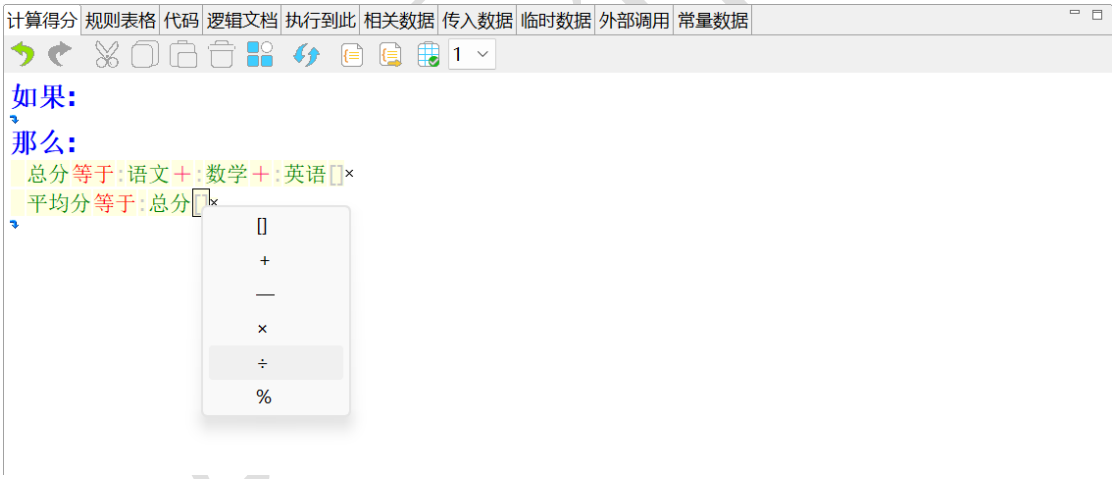
用上一步算出的“总分”除以科目数得出“平均分”，点击“请点击选择[]×”选择“选择值”下的“临时数据”中的总分：



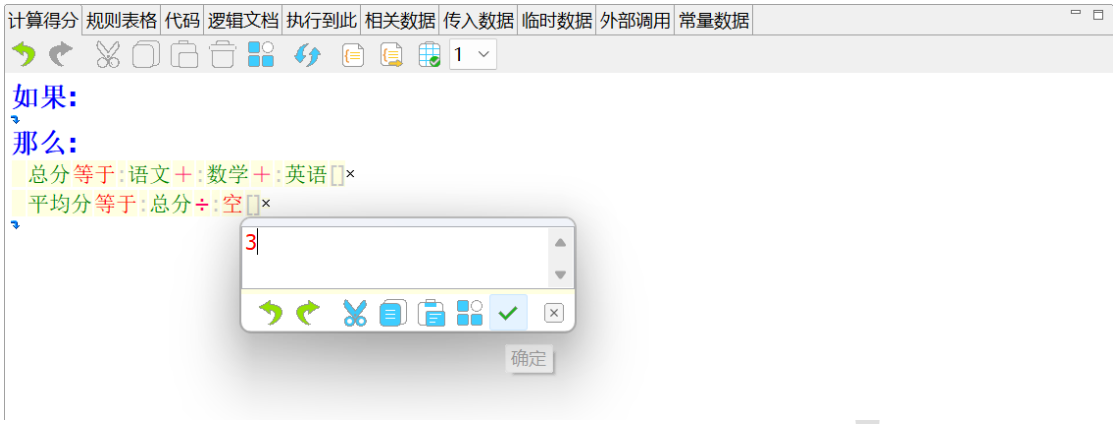
（也可以右键“总分”复制，然后粘贴到“请点击选择[]×”）



点击“总分”后面的“[]”，选择“数值间操作符”，这里我们是要求平均值，因此选择“÷”：

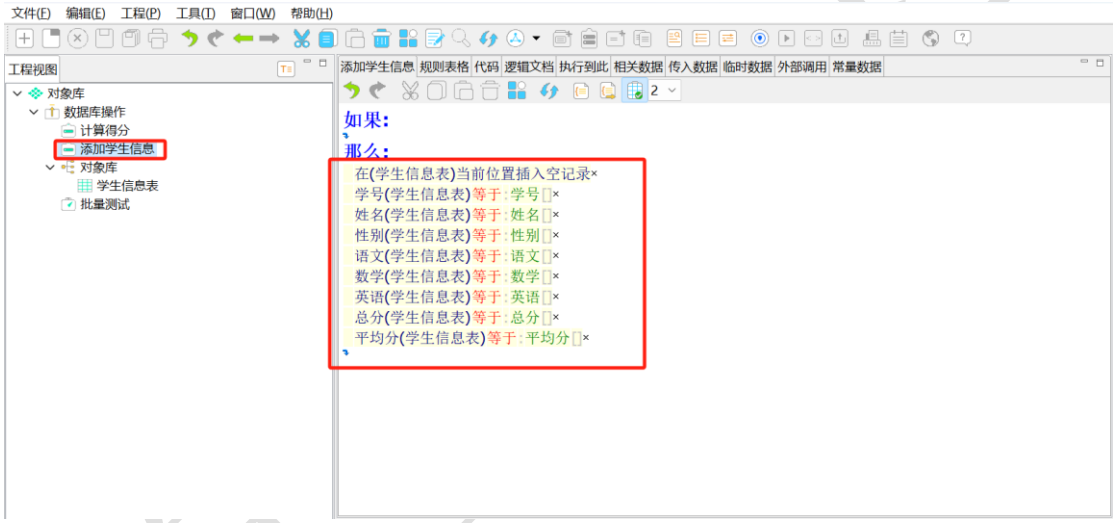


点击请点击选择[]×选择“输入常量值”，出现空[]，点击“空”在弹出框中输入科目数：



(2) 保存数据到计算表格

与上面“学号”、“姓名”、“性别”赋值到“学生信息表”相同，这里将“语文”、“数学”、“英语”、“总分”、“平均分”分别赋值：



以上步骤完成之后，点击工具栏中的保存数据就可以进行测试了，测试方法同上。

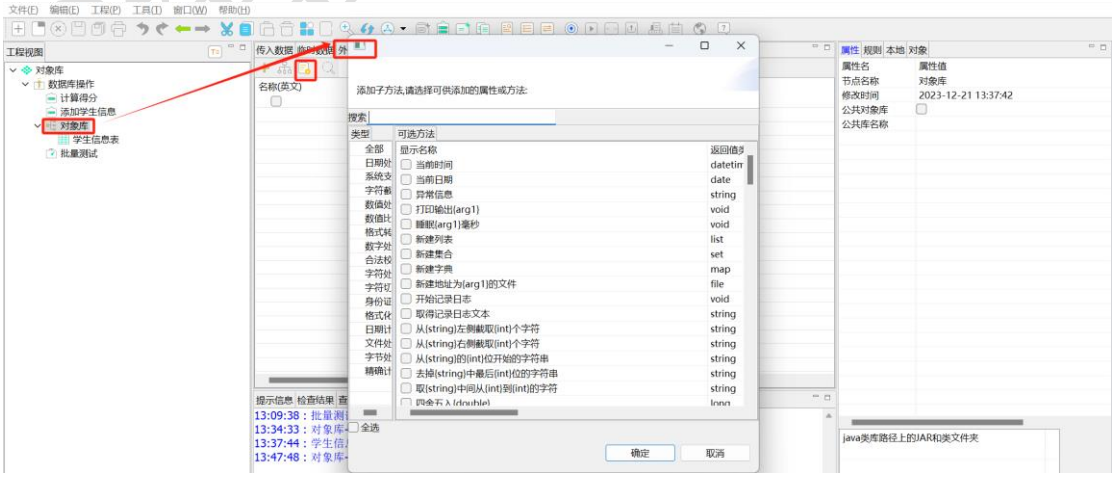
2.2.3. 外部调用

在规则引擎中，我们将一些常用的方法进行了封装，可通过“添加公式”进行调用，根据不同类型又将这些方法进行了分门别类，总共分为 16 类。而其他未做封装的可以通过“添加”输入包名进行搜索调用。

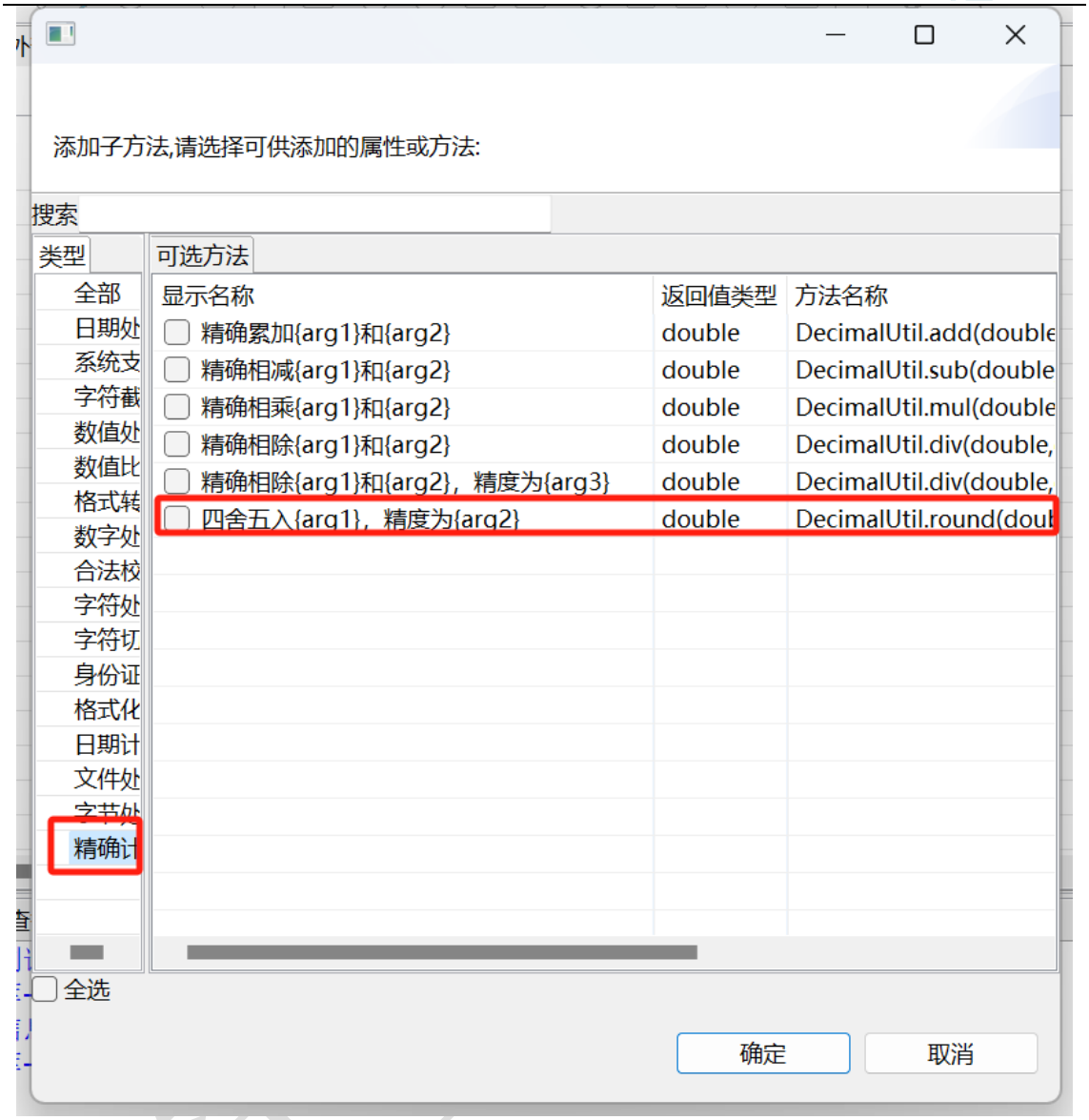
这里我们只取其中的一个进行调用，由于上面计算所得到的“平均分”可能有很多位小数，而实际中分数往往是通过四舍五入取小数点后一位，因此这里我们通过调用类型“精确计算”中的“四舍五入{arg1}，精度为{arg2}”来取整。

1. 添加外部调用公式

点击“对象库”中“外部调用”，在工具栏中通过  来“添加公式”：

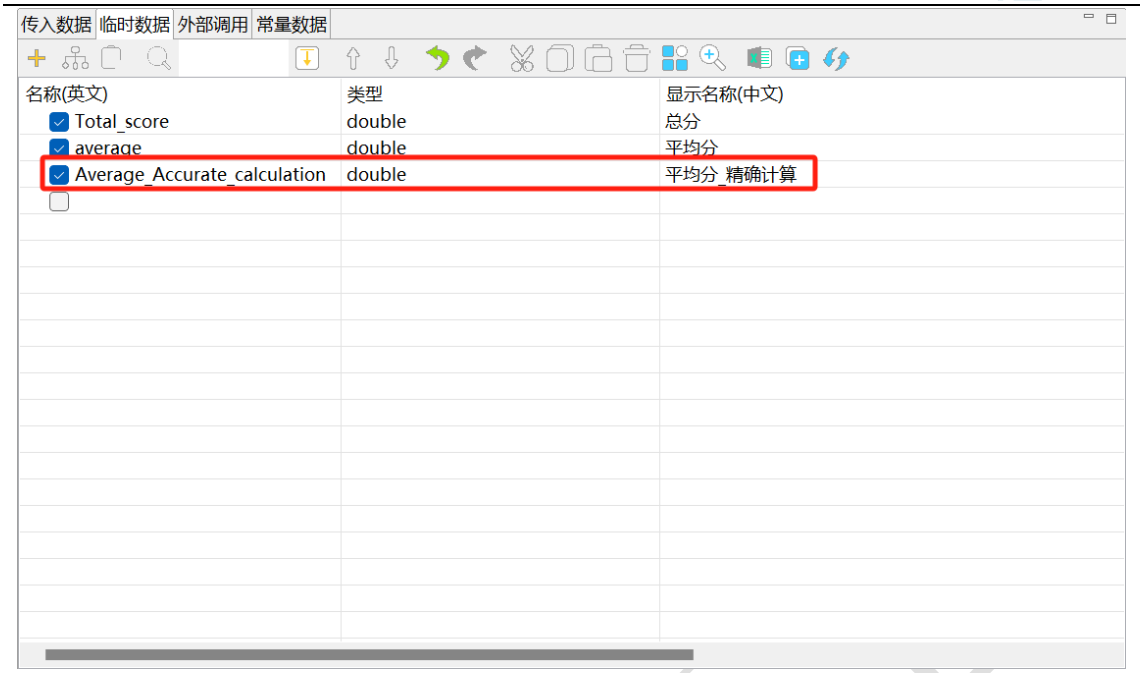


选择类型为“精确计算”在右侧“可选方法”中勾选“四舍五入{arg1}，精度为{arg2}”



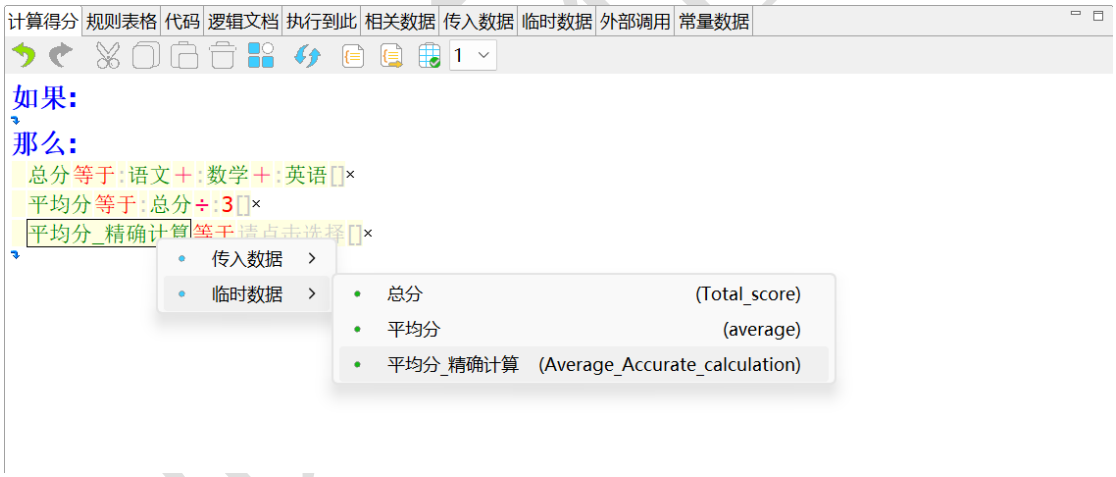
2. 添加临时数据

在“临时数据”中添加一个“平均分_精确计算（Average_Accurate_calculation）”字段：

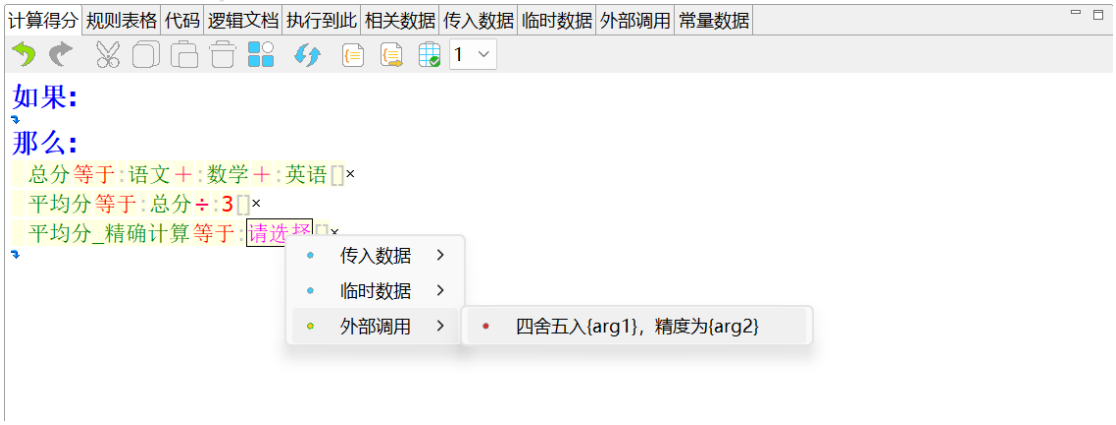


3. 在规则中调用

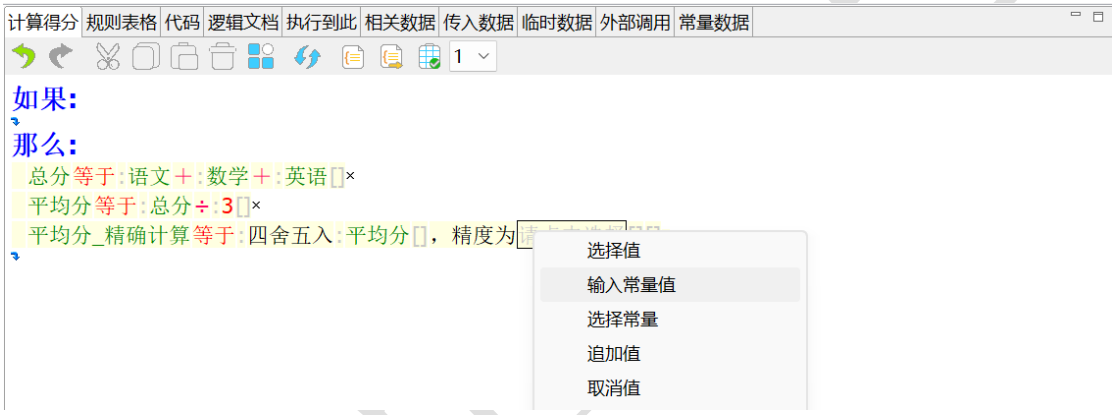
点击“计算得分”规则，点击选择“变量赋值”“临时数据”中的“平均分_精确计算（Average_Accurate_calculation）”：



点击请点击选择选择“选择值”“外部调用”中的“四舍五入{arg1}，精度为{arg2}”：

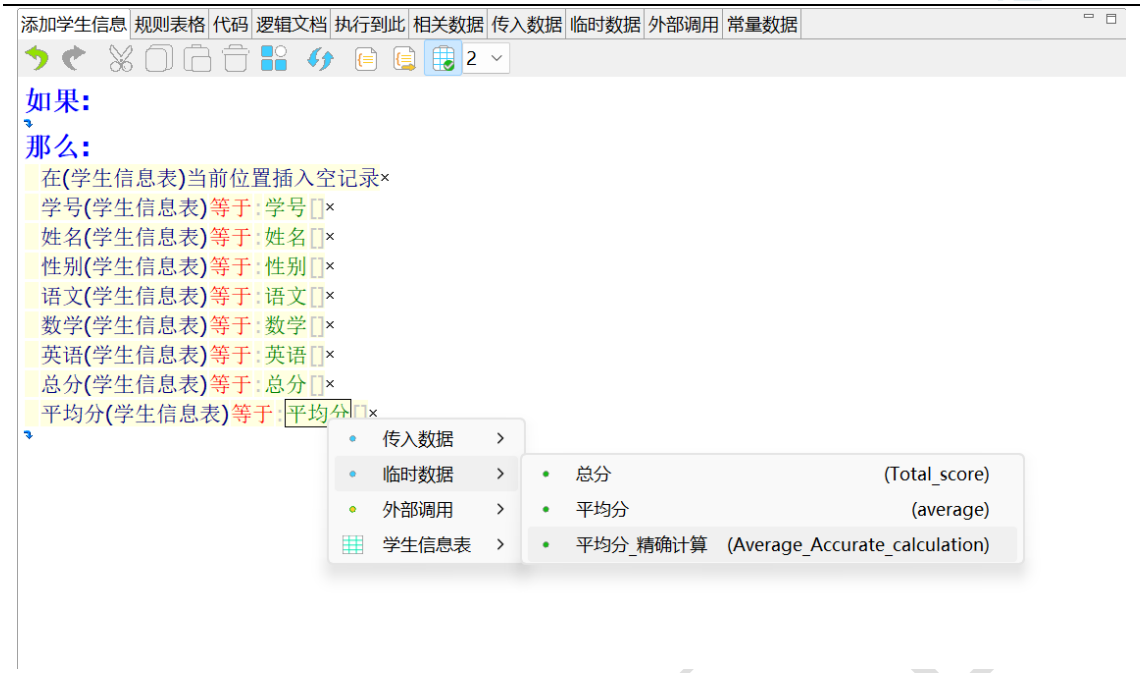


这里第一个{arg1}选择“临时数据”中的“平均分”，第二个{arg2}为要精确到的小数位数（1 表示保留一位小数，2 表示保留两位小数，以此类推）选择“输入常量值”，在弹出框中输入 1：



4. 回传值

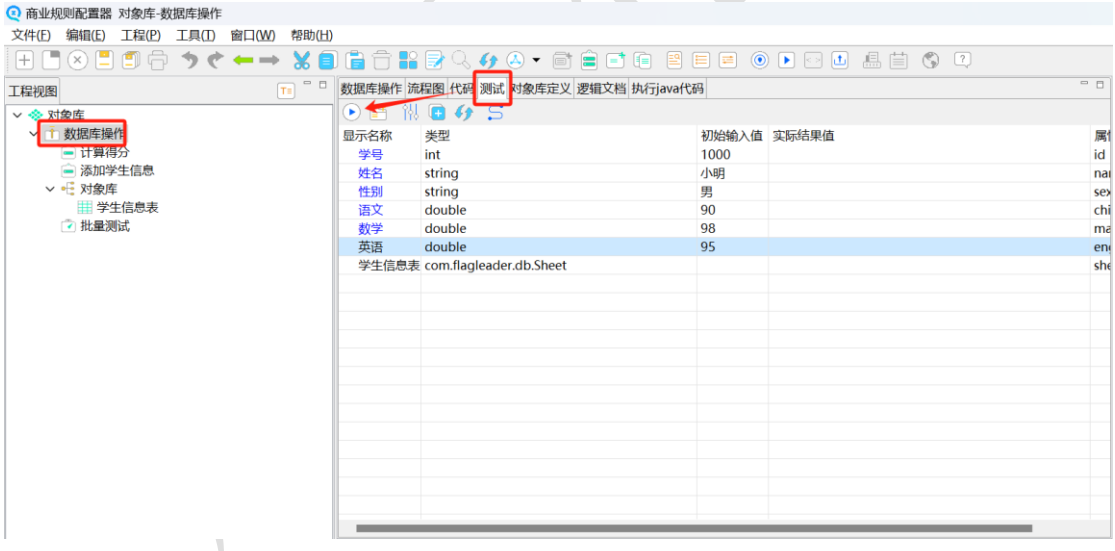
将“添加学生信息”规则中“平均分（学生信息表）”的赋值由“临时数据”中的“平均分”更改为“平均分_精确计算”：



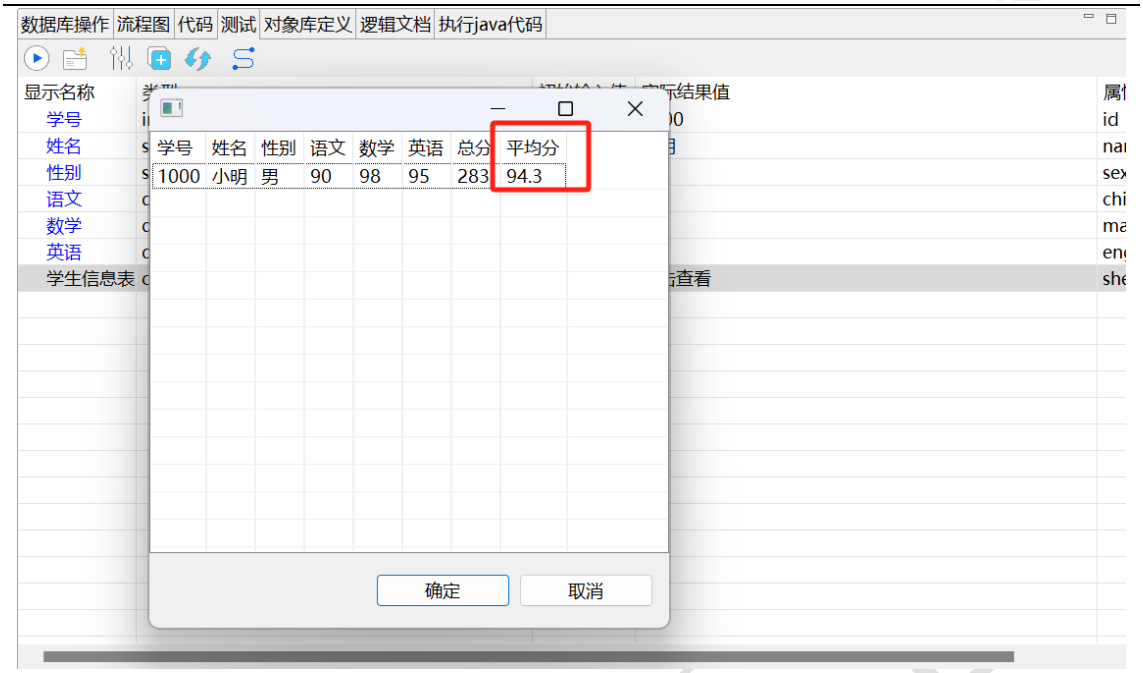
完成之后点击保存。

5. 测试

点击“数据库操作”规则包，选择“测试”：



查看“学生信息表”，两次“平均分”数值比较：



2.2.4. 常量数据

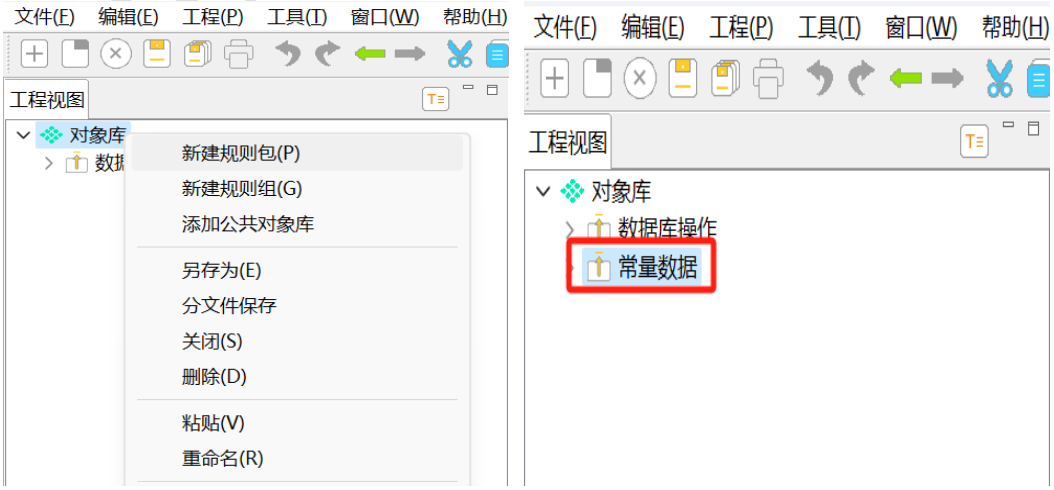
在规则中当我们要给变量赋予常量值时，我们可以通过直接输入常量值来赋予，也可以将此值添加在常量数据中，再将此常量数据中的字段赋给该变量从而获取常量值。

相较于直接输入常量值，添加常量数据的优势在于：当有多条变量的值是同一常量时，如需更改此常量，传统的直接输入常量值需要逐条的进行更改。而将该常量添加到常量数据中再进行赋值则只需要更改常量数据中的数值就可完成全部更改，这样就提高了开发效率。

例如，在“关联决策表”中有学生姓名和年级两个字段，现需统一修改年级信息，就可将此年级添加至常量数据，通过修改常量数据完成年级信息更改。具体规则过程如下：

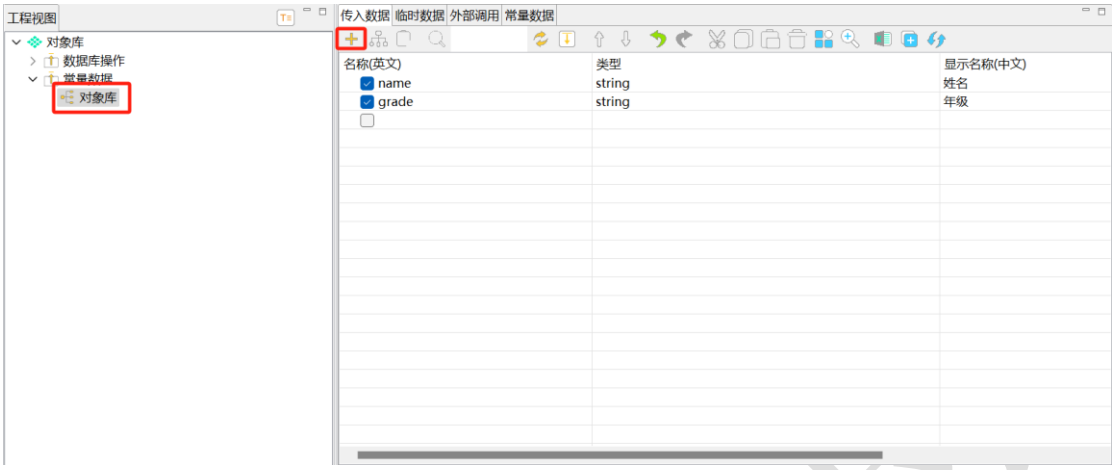
1. 新建规则包

在“对象库”工程下右键“新建规则包”，并命名为“常量数据”：

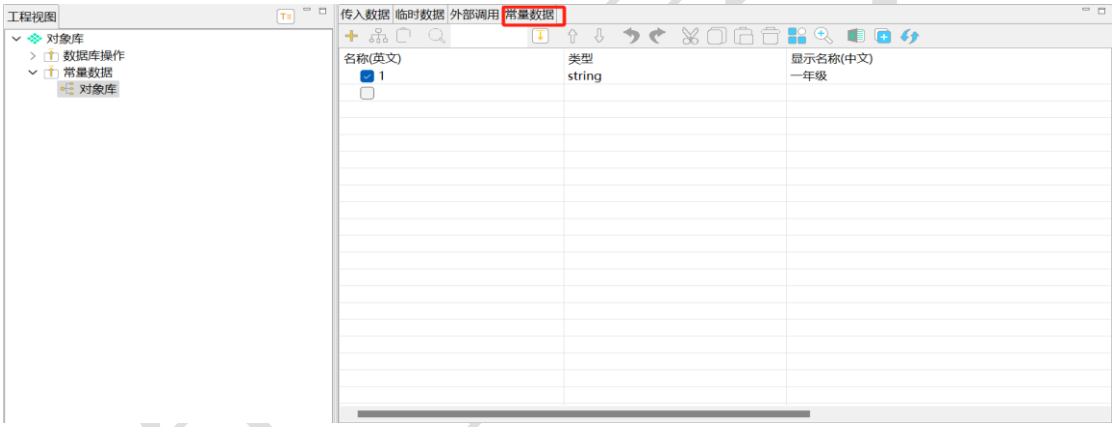


2．定义对象库变量

打开“常量数据”规则包，在对象库传入数据中添加“姓名（name）”和“年级（grade）”两个变量，如下图所示：

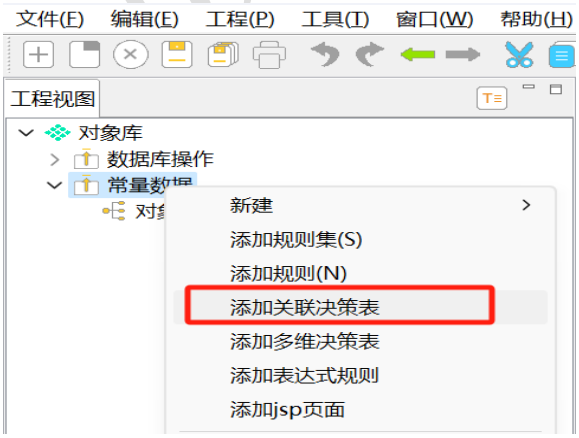


在对象库常量数据中添加年级数据，操作如下图所示：

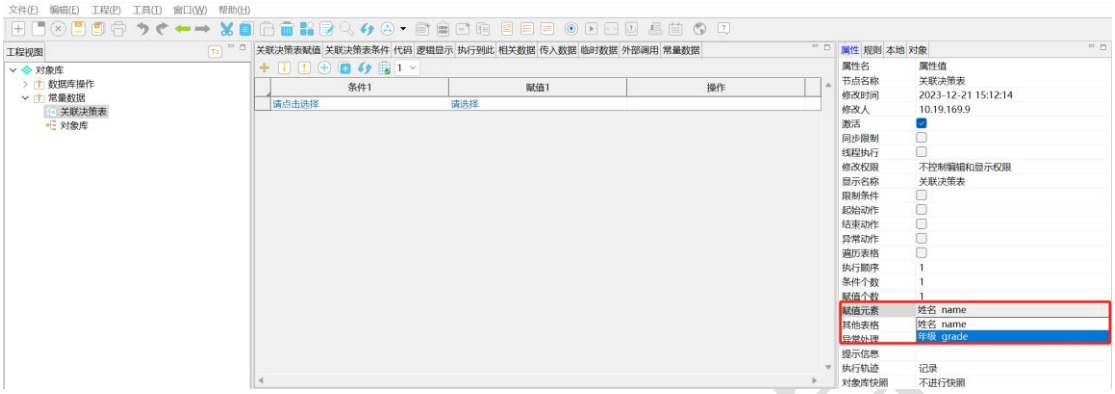


3．添加关联决策表

对象库变量添加完成之后需要在规则包下添加关联决策表。选择“常量数据”规则包，右键“添加关联决策表”：

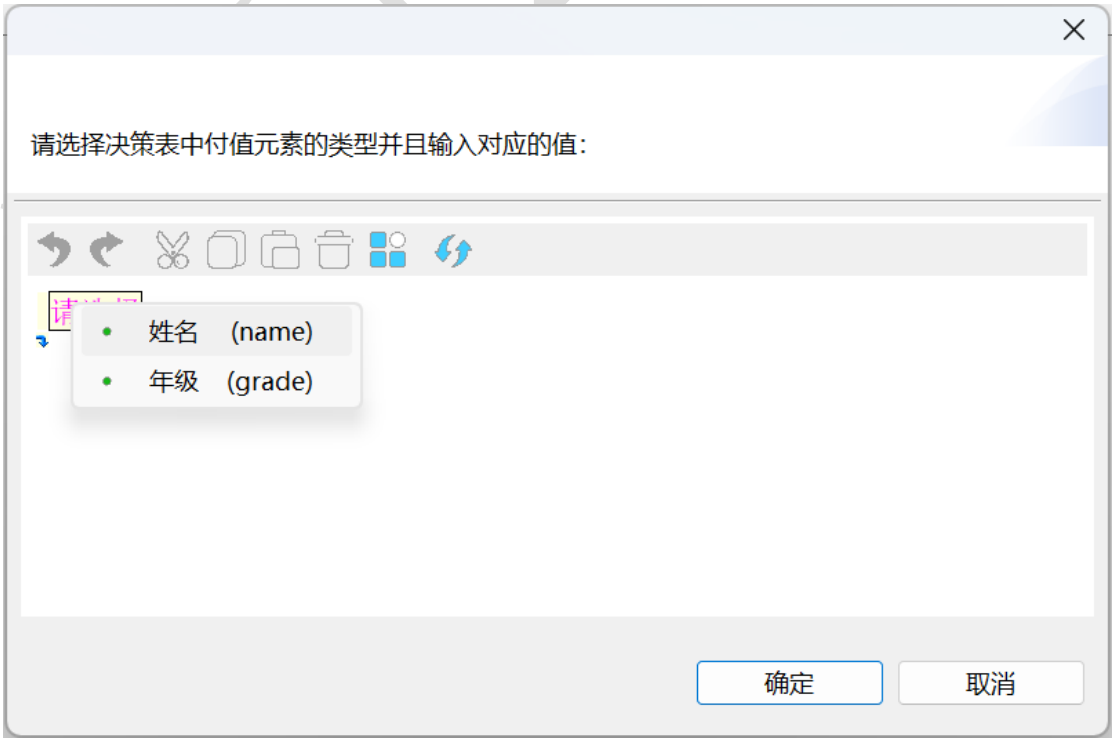


选择“关联决策表”，在右侧属性窗口中，将属性名“赋值元素”对应的属性值改为“年级 grade”：

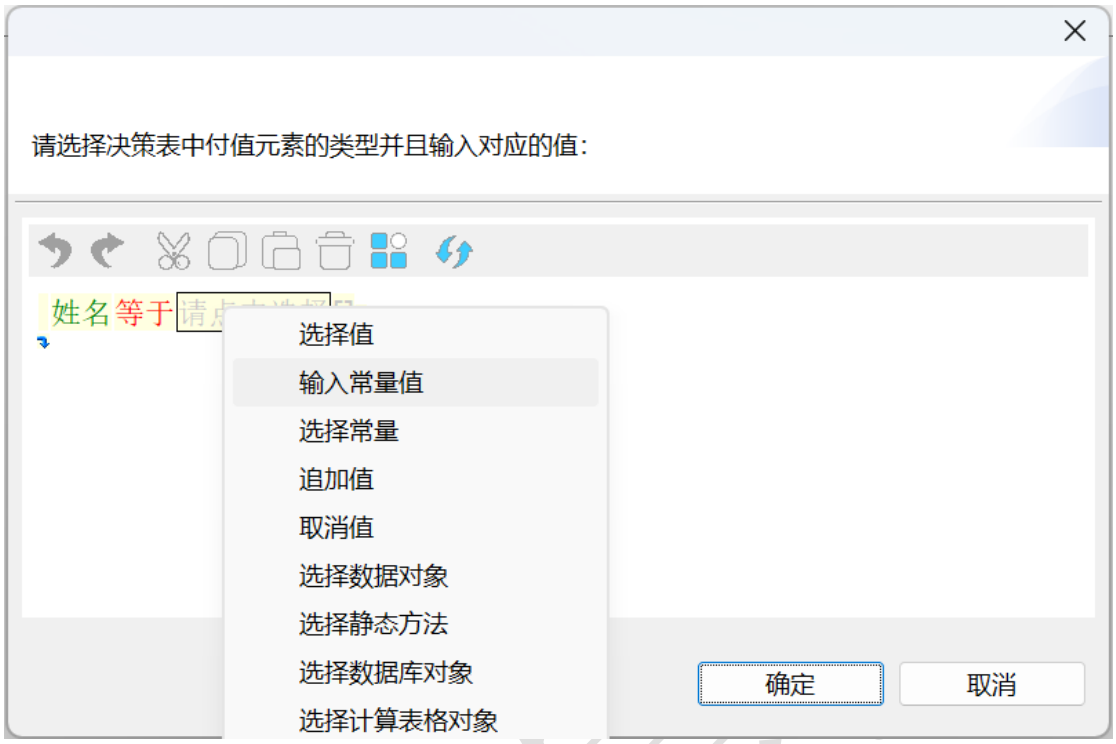


在编辑窗口中添加条件列，并赋值：

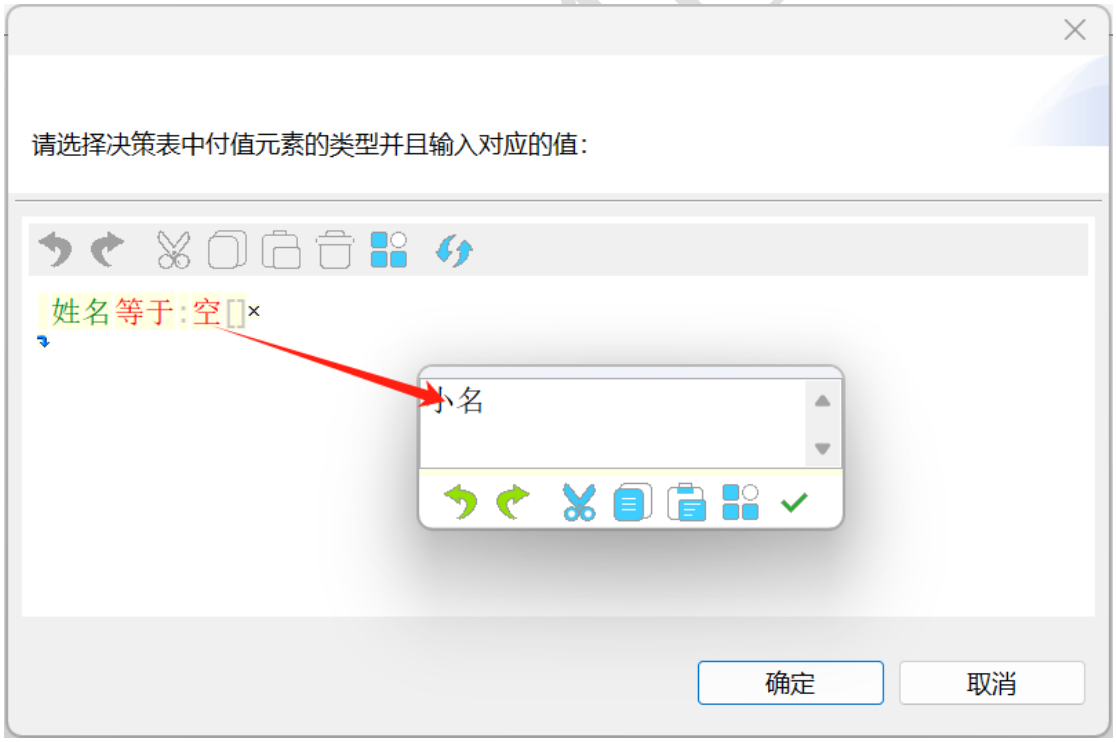
在弹出页面中添加条件并赋值。点击“ ”添加条件，选择姓名：



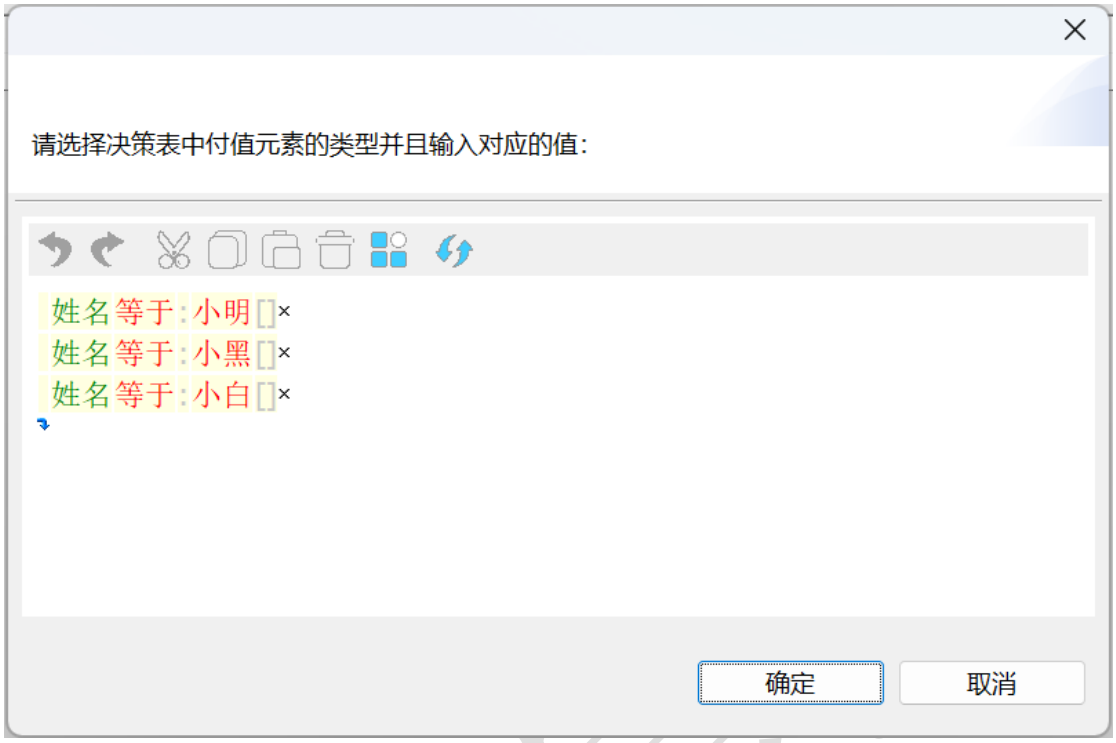
对“姓名”进行赋值。点击“请点击选择[]”→“输入常量值”：



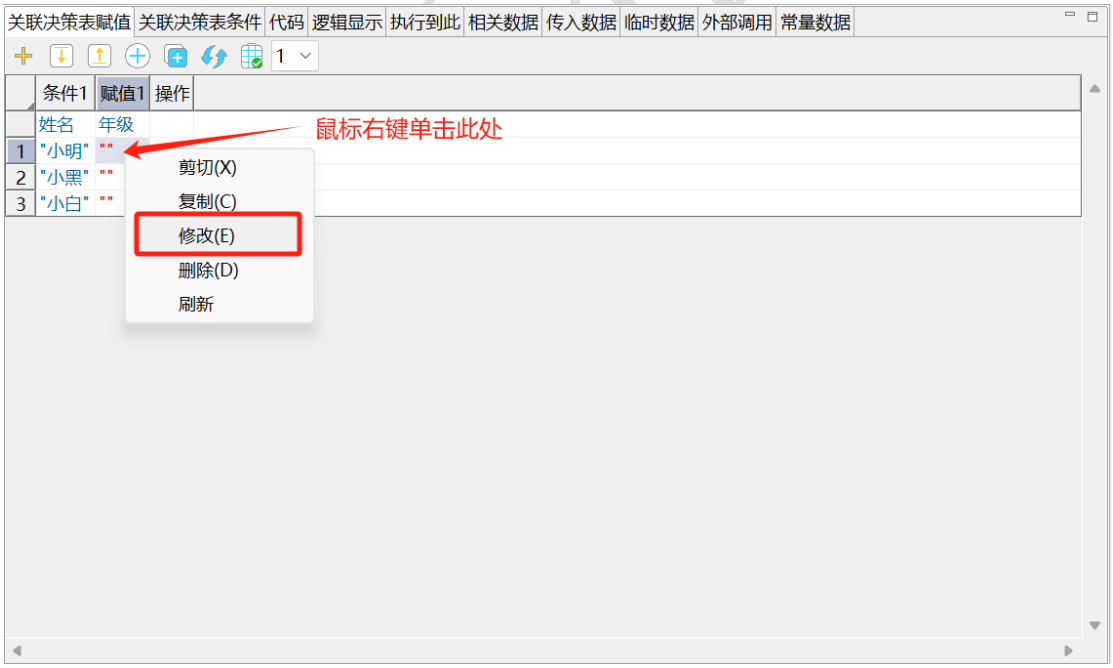
点击“空”弹出输入框，在输入框中输入姓名：



根据上述方法在再次添加两条数据，完成后结果如下图所示：

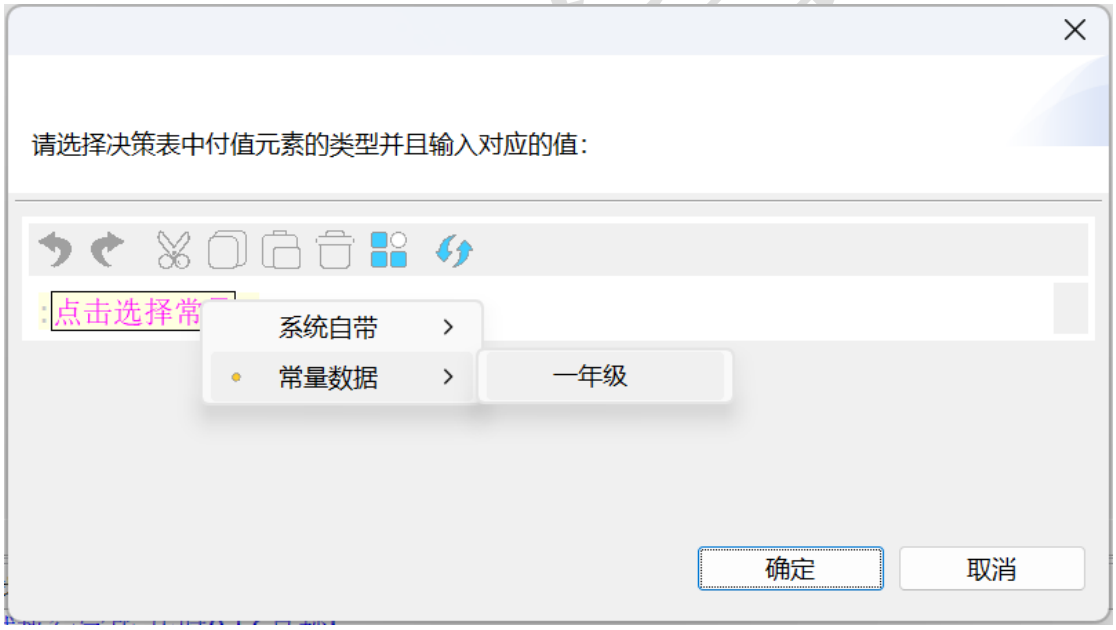


完成之后点击确定，然后对“赋值元素”年级进行赋值。右键点击“年级”下方的单元格，选择修改：

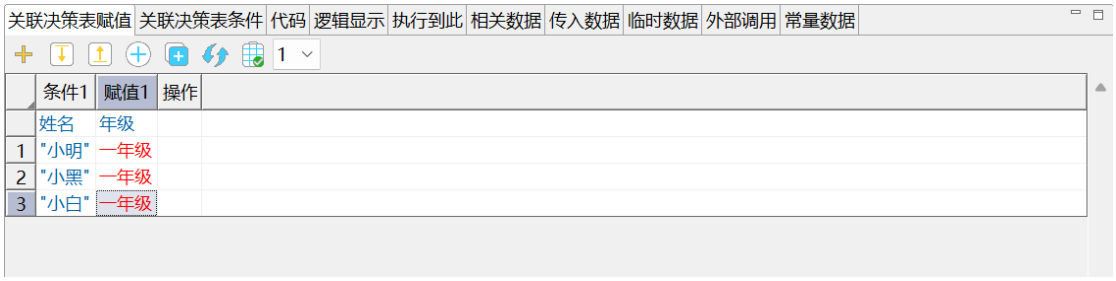


在弹出框中点击“:”选择值类型，选择“选择常量”：

点击“点击选择常量”出现“系统自带”和“常量数据”。其中“系统自带”包含规则名、规则包执行名、规则包显示名等，选择某一个量就可获得其相应的值，而“常量数据”就是我们在对象库常量数据中添加的值。这里我们选择“常量数据”：



根据上述方法步骤，对另外两条数据赋值，完成后显示如下：



4. 保存并编译

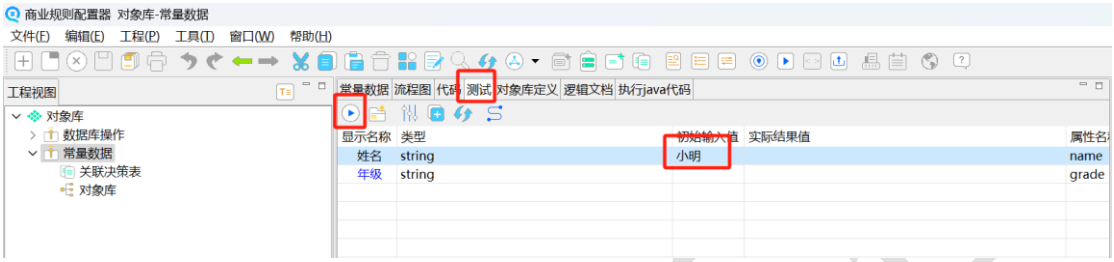
点击工具栏中的“ ”保存并编译规则包：

保存之后在消息窗口中出现如下提示信息：

其中 rpk 文件是规则包源文件，可直接通过规则配置器来打开，rsc 文件是规则编译后的二进制文件。若出现红色提示信息则表示程序报错。

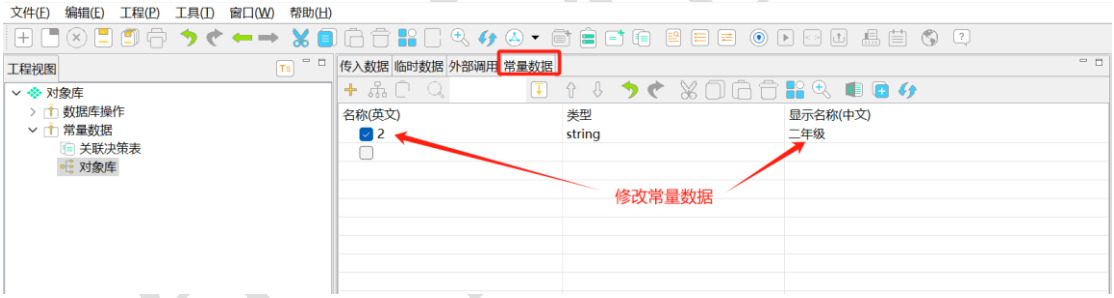
5．测试

规则保存编译之后需对其进行测试，选择规则包，在编辑窗口中切换至“测试”选项卡，在“初始输入值”中输入传入数据，点击“ ”执行，在“实际结果值”下回输出测试结果数据。过程如下图所示：



6．修改常量数据

修改年级信息，将“关联决策表”中的年级由“一年级”改为“二年级”。在对象库的常量数据中修改常量值：

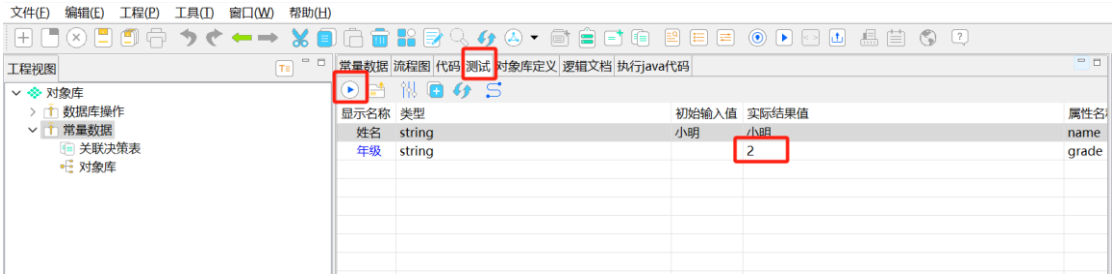


修改之后“关联决策表”的值与修改前“关联决策表”的值对比：

修改后

修改前

点击“ ”保存并编译，然后在测试中进行测试，过程如下：



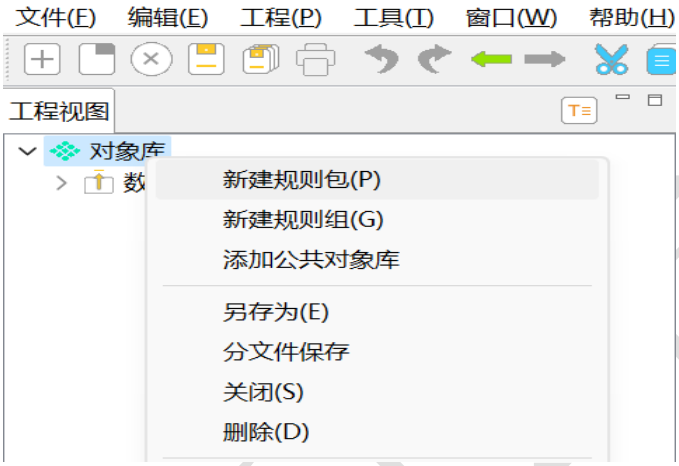
2.3. 对象库下

2.3.1. 数据库

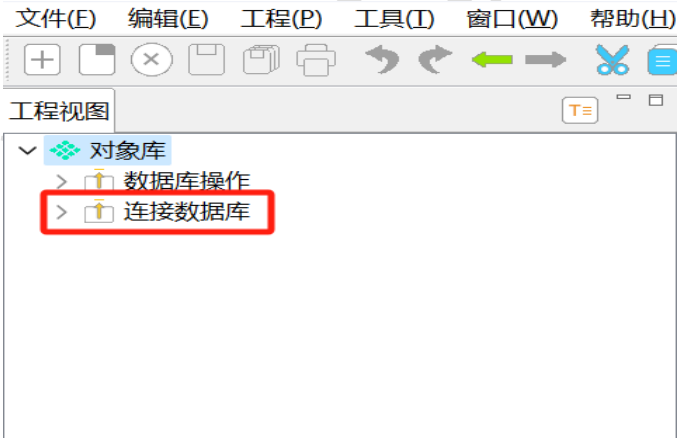
2.3.1.1. 数据库表

1. 新建规则包

选中工程名为“对象库”的工程，右键点击“对象库”，点击菜单项中的“新建规则包”，如下图：



并将其命名为“连接数据库”，显示如下：



“连接数据库”规则包创建完成。

2. 连接数据库

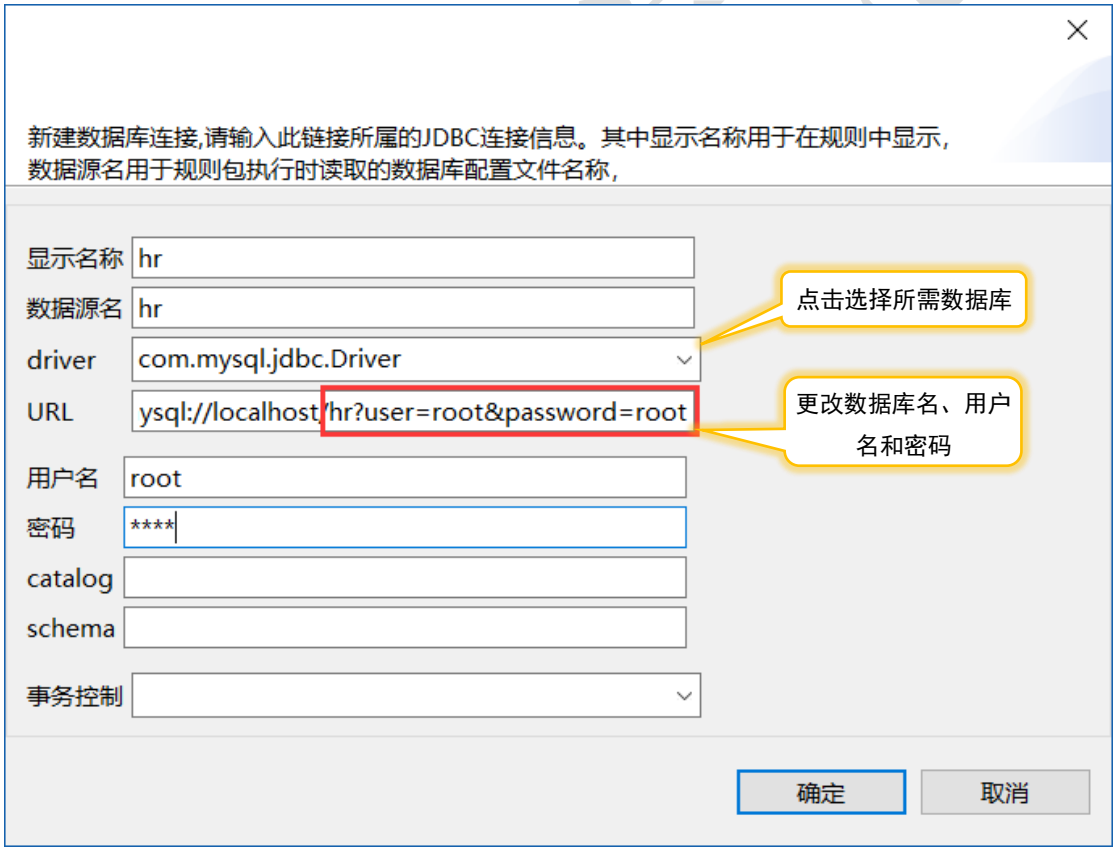
连接到数据库有三种方法分别是：“添加数据库连接”、“从 dbs 文件中导入表”、“添加 mongo 数据库”。

(1) 添加数据库连接

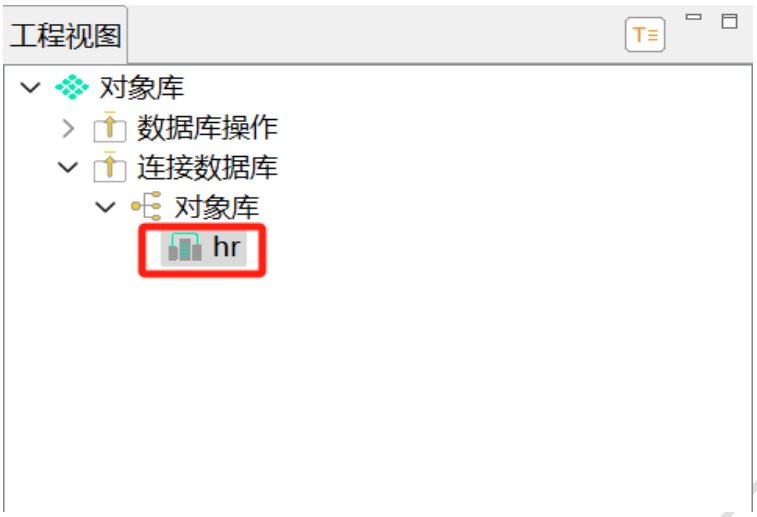
将“连接数据库”规则包点开，可以看到“对象库”，右键点击对象库，选择菜单项中

的“添加数据库连接”，如下图：

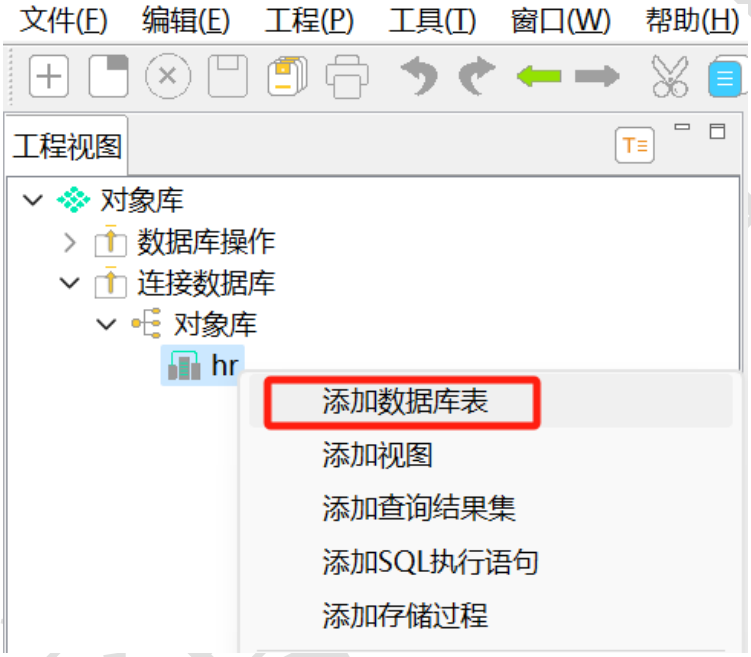
在弹出窗体中依次填入以下信息（这里以 mysql 数据库为例）：



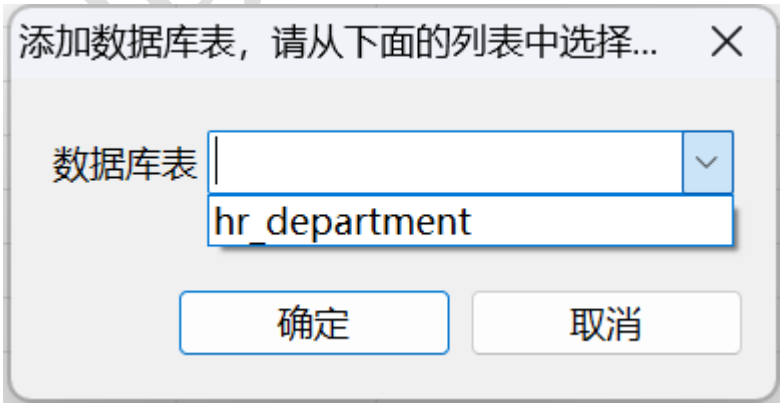
完成后点击确定，在对象库下出现数据库。



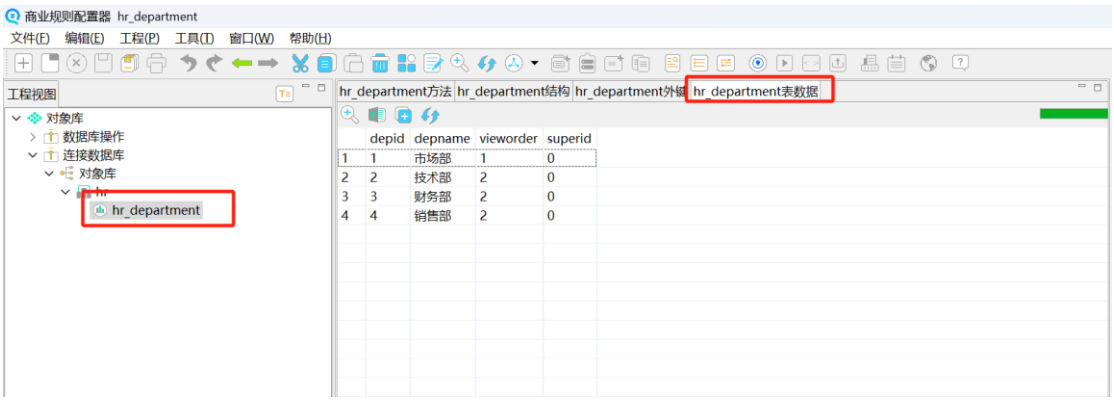
右键“hr”数据库，选择“添加数据库表”。



在弹出框中选择“hr_department”



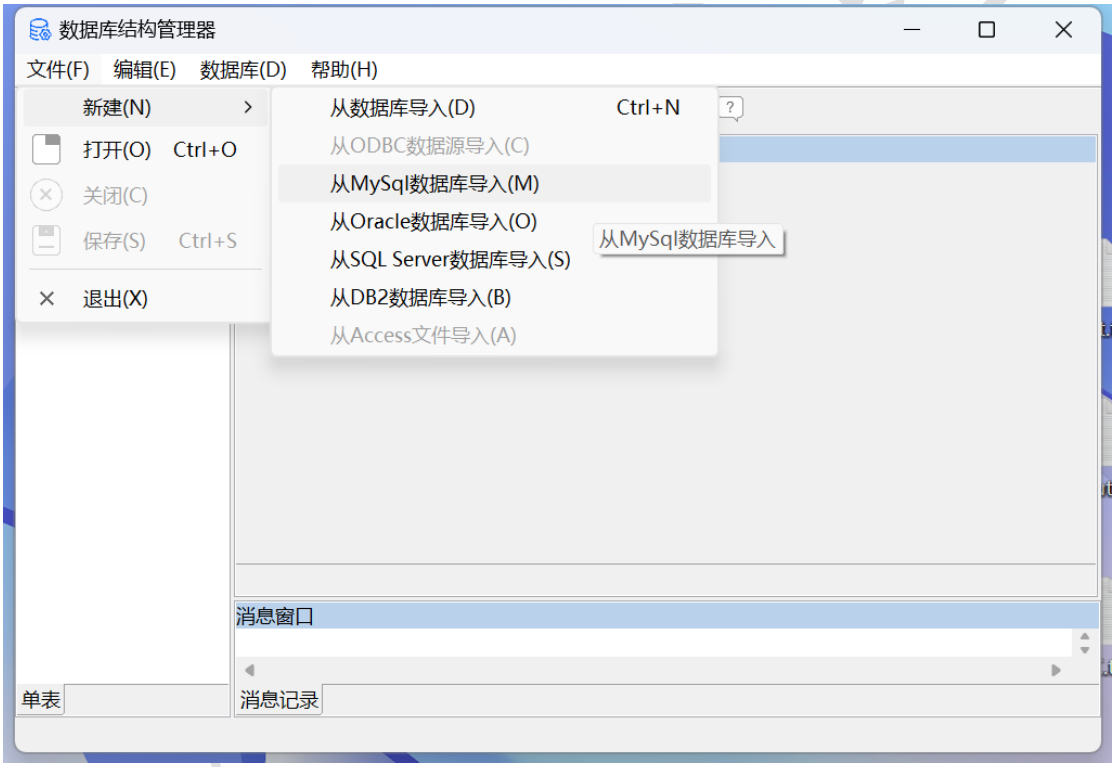
选好后点击“确定”， 点击“hr_department”，查看“hr_department 表数据”菜单项，显示如下：



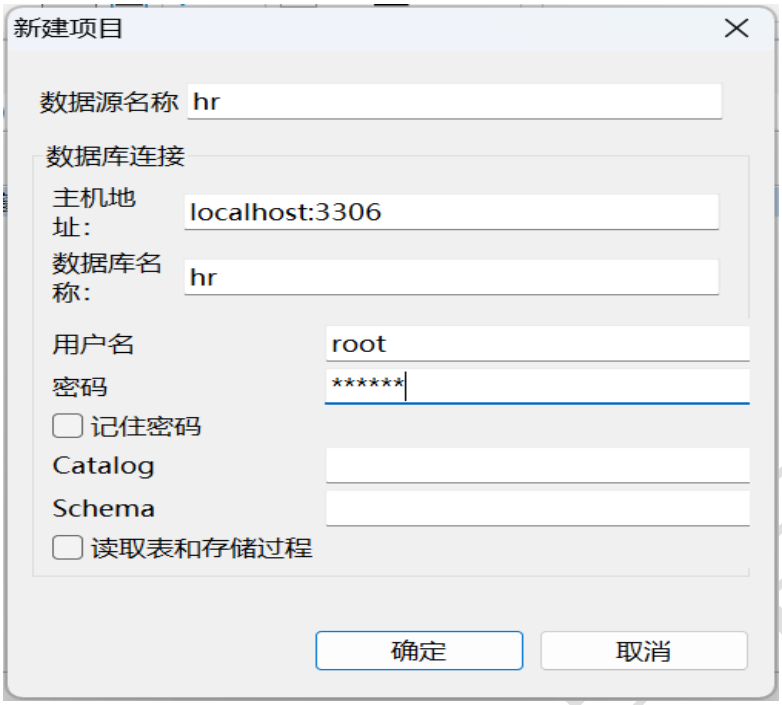
(2) 从 dbs 文件中导入表

要从 dbs 文件导入首先就要生成 dbs，这里我们要用到“数据库配置器”将 SQL 文件转换为 dbs。

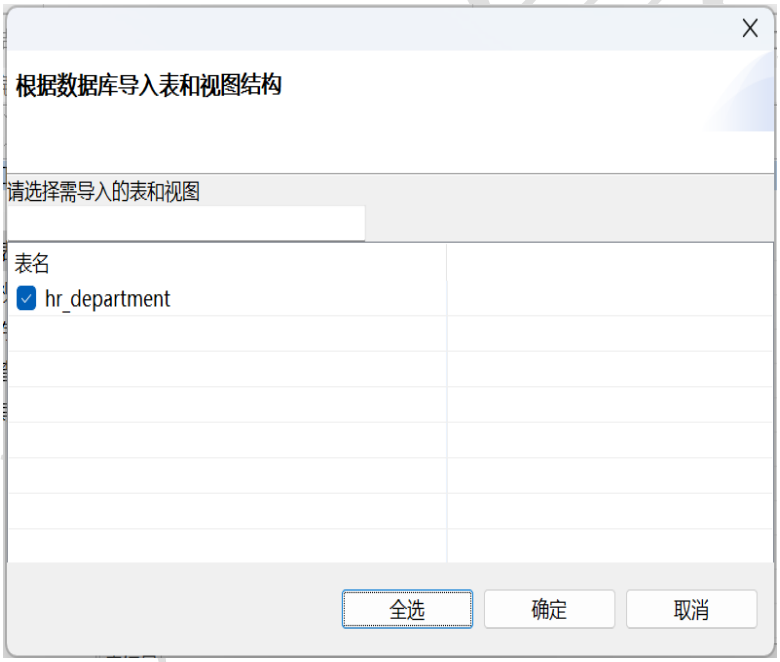
打开“数据库配置器”，点击左上角“文件”，在“新建”下拉列表中选择自己的数据库，以 mysql 为例：



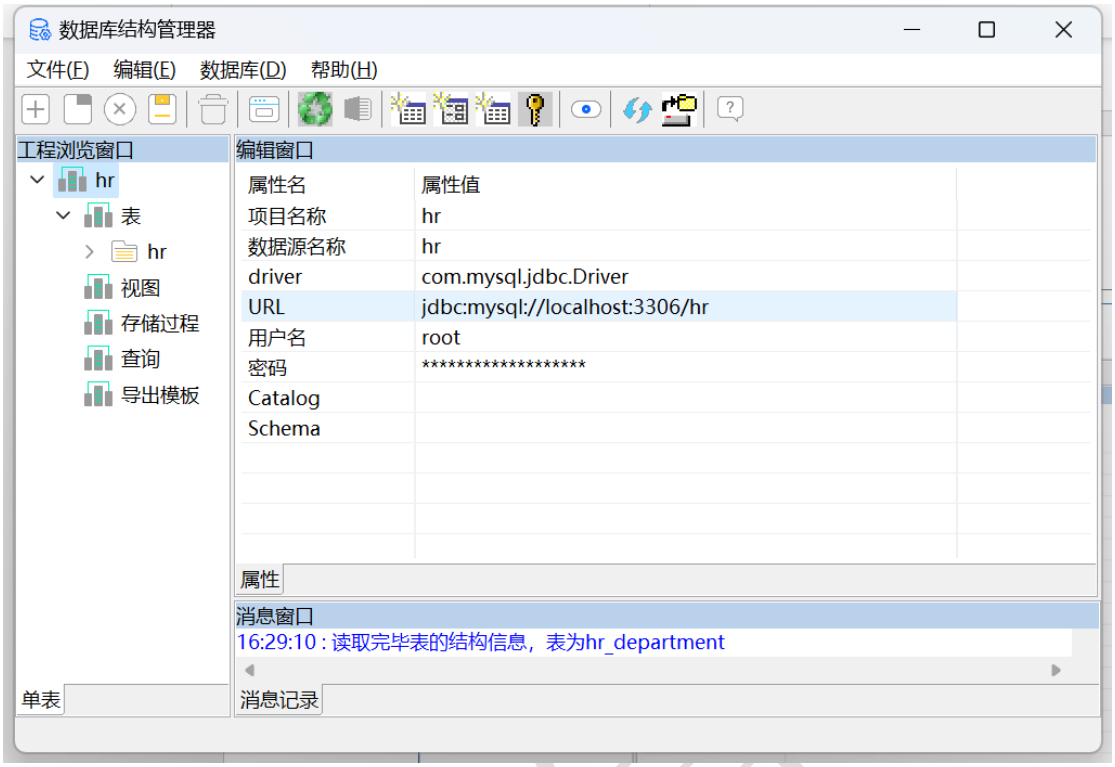
依次填入下列信息：



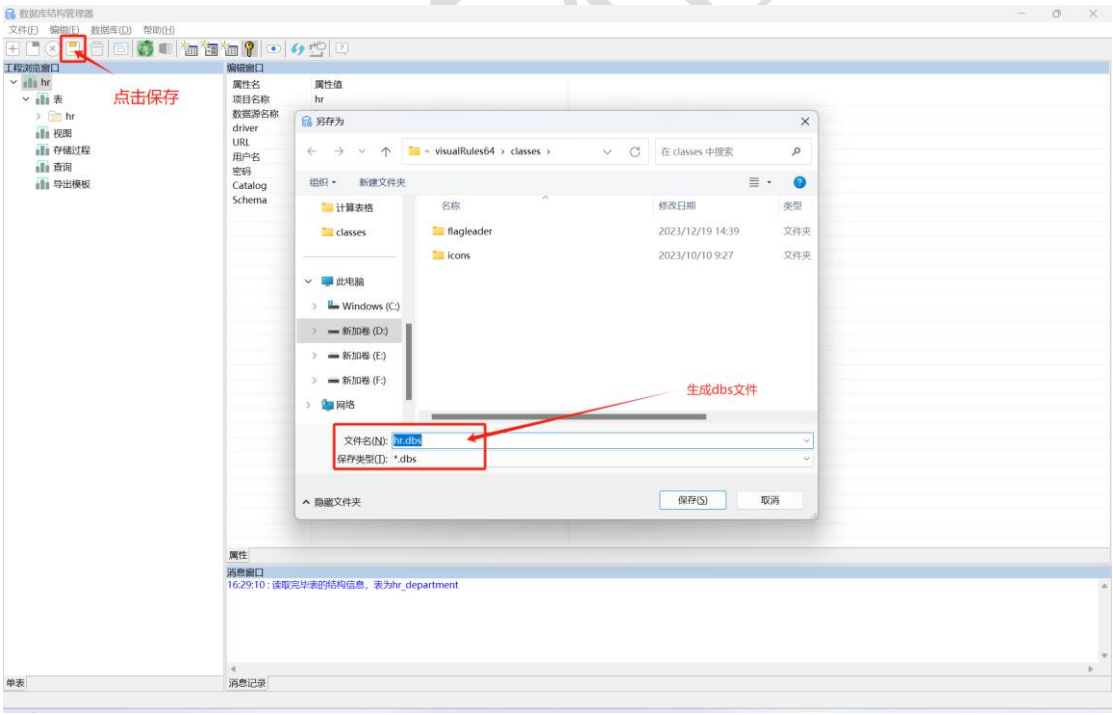
完成后点击确定，在弹出框中选择要导入的表、视图、存储过程在前面勾选。



添加完成如下图：

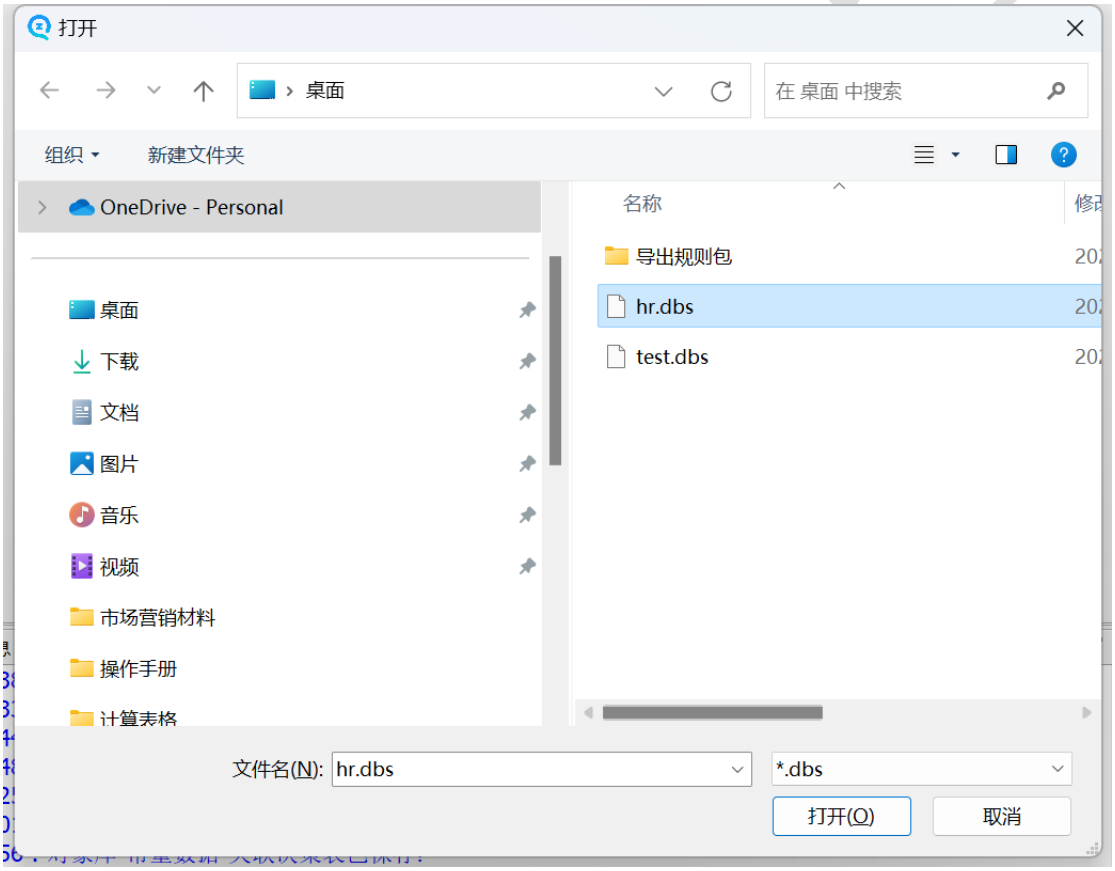


点击“保存项目”生成 dbs 文件。



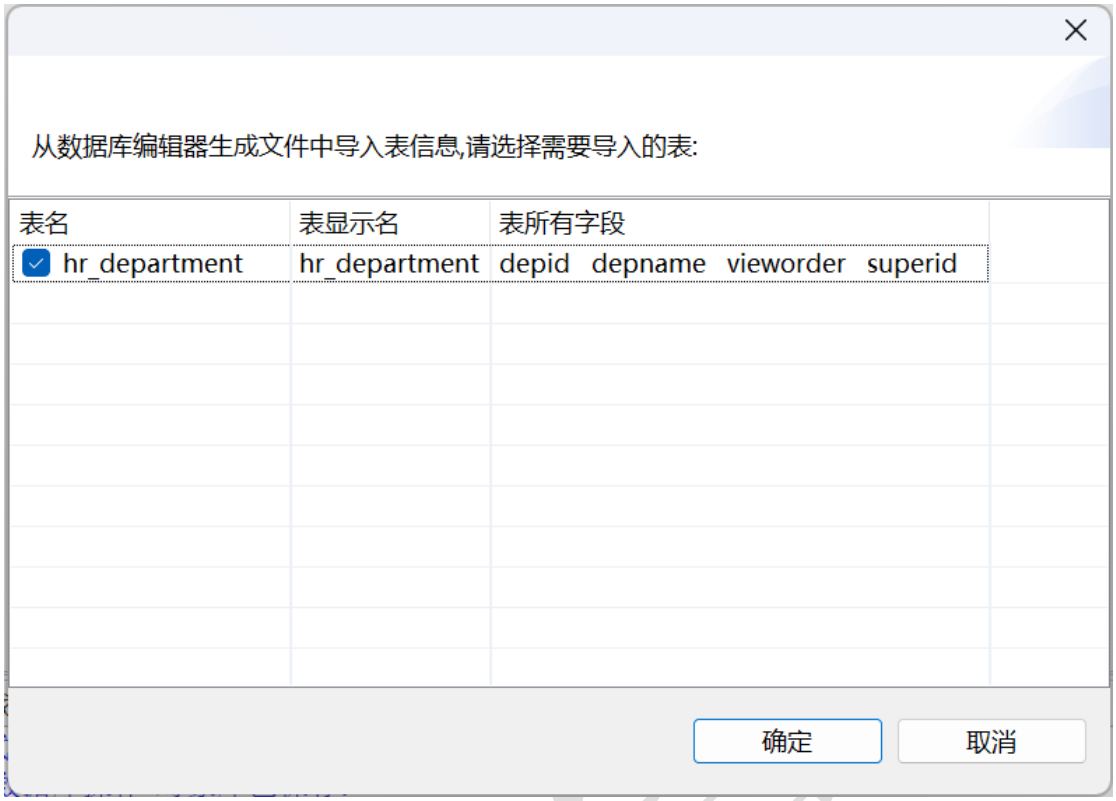
回到“规则配置器”，在“连接数据库”规则包的对象库下右键，选择“从 dbs 文件中导入表”：

选择刚刚生成的 dbs 文件，点击打开：



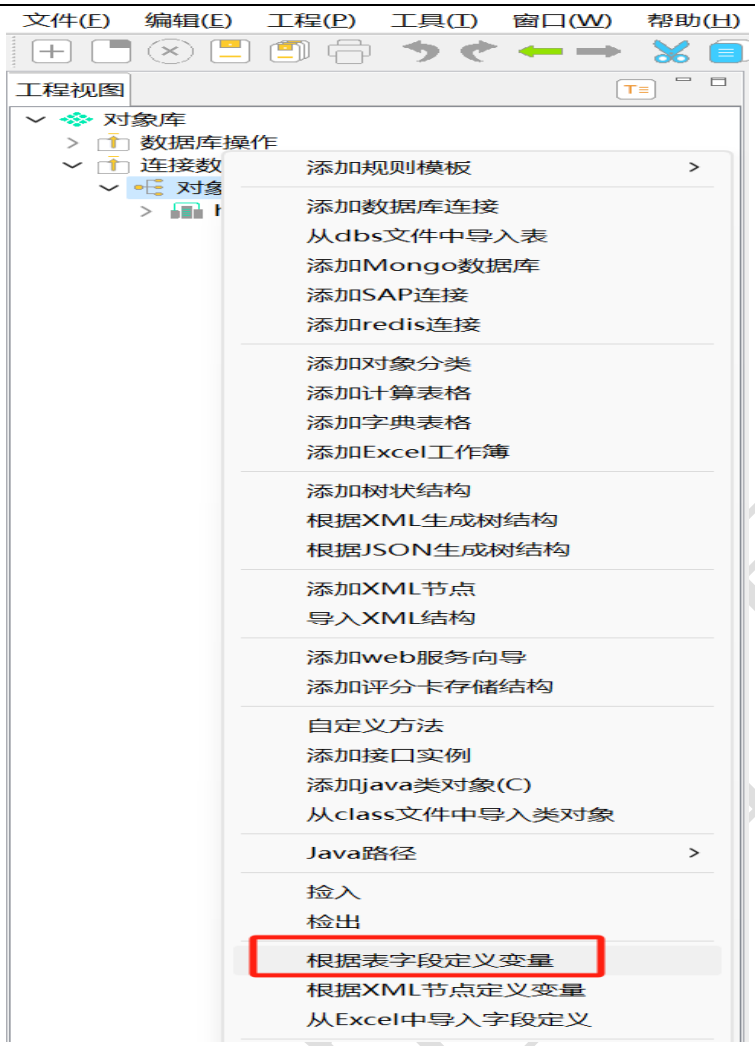
勾选需要导入的表：

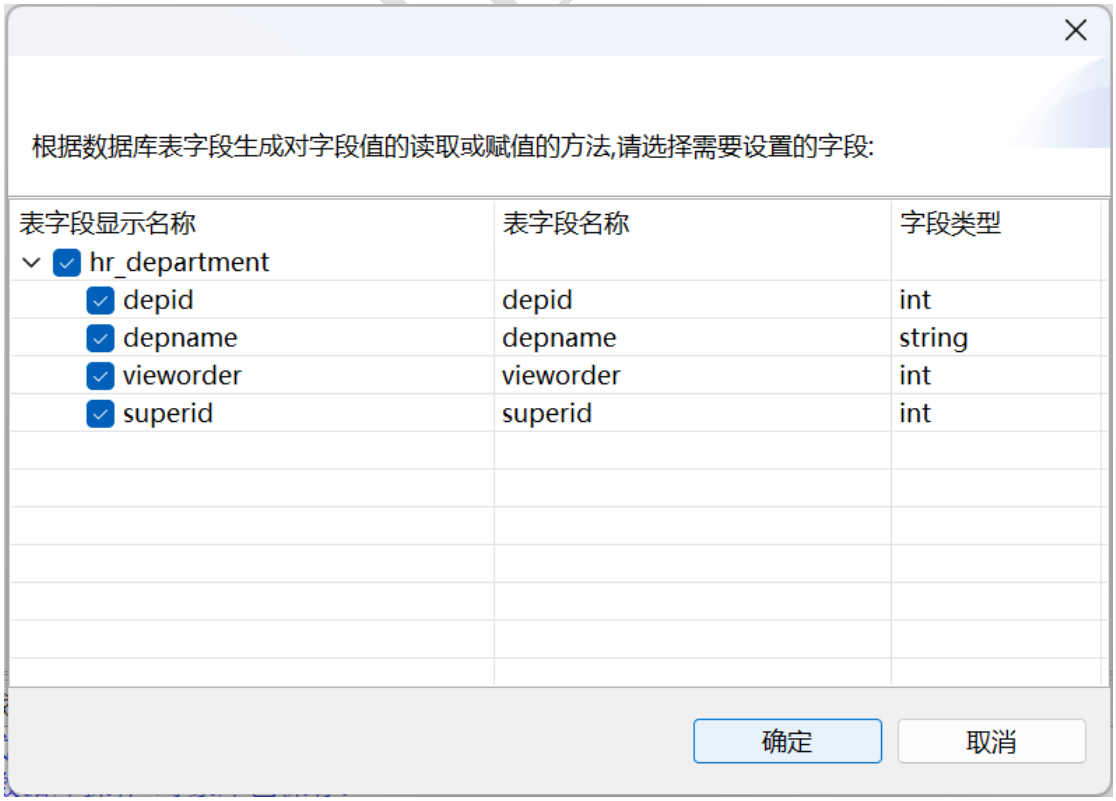
\



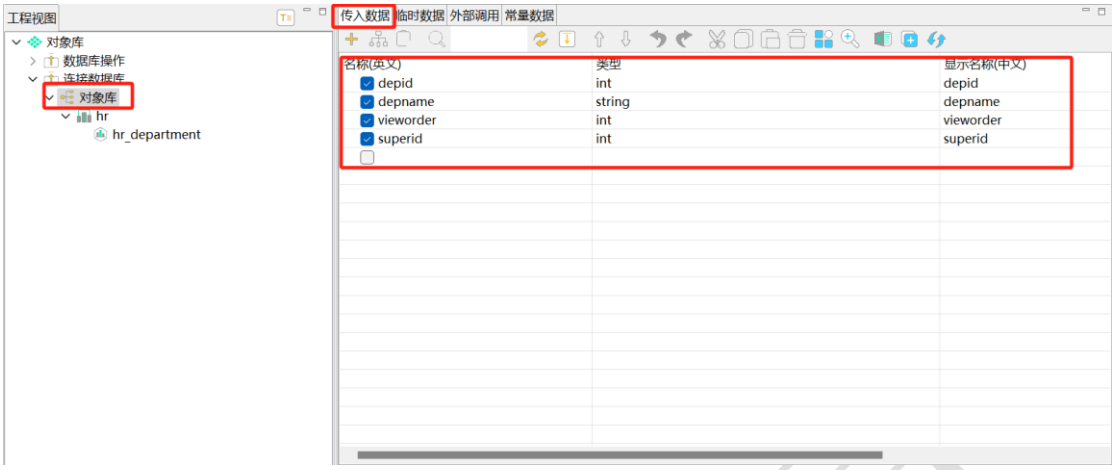
3．创建对象库变量

在对象库中添加“hr”数据库中的“hr_department”表中的字段。右键对象库，选择“根据表字段定义变量”：


文件(F)
编辑(E)
工程(P)
工具(T)
窗口(W)
帮助(H)
工程视图
对象库
数据库操作
连接数
对象
添加规则模板
添加数据库连接
从dbs文件中导入表
添加Mongo数据库
添加SAP连接
添加redis连接
添加对象分类
添加计算表格
添加字典表格
添加Excel工作簿
添加树状结构
根据XML生成树结构
根据JSON生成树结构
添加XML节点
导入XML结构
添加web服务向导
添加评分卡存储结构
自定义方法
添加接口实例
添加java类对象(C)
从class文件中导入类对象
Java路径
检入
检出
根据表字段定义变量
根据XML节点定义变量
从Excel中导入字段定义


根据数据库表字段生成对字段值的读取或赋值的方法,请选择需要设置的字段:
表字段显示名称
表字段名称
字段类型
hr_department
depid
depid
int
depname
depname
string
vieworder
vieworder
int
superid
superid
int
确定
取消

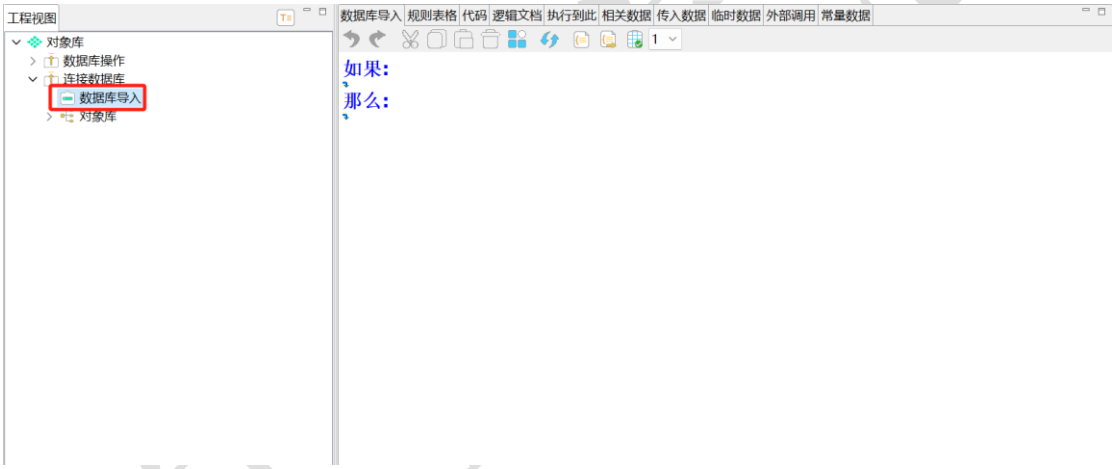
打开对象库，根据表字段生成的对象库变量如下：



4．编写规则

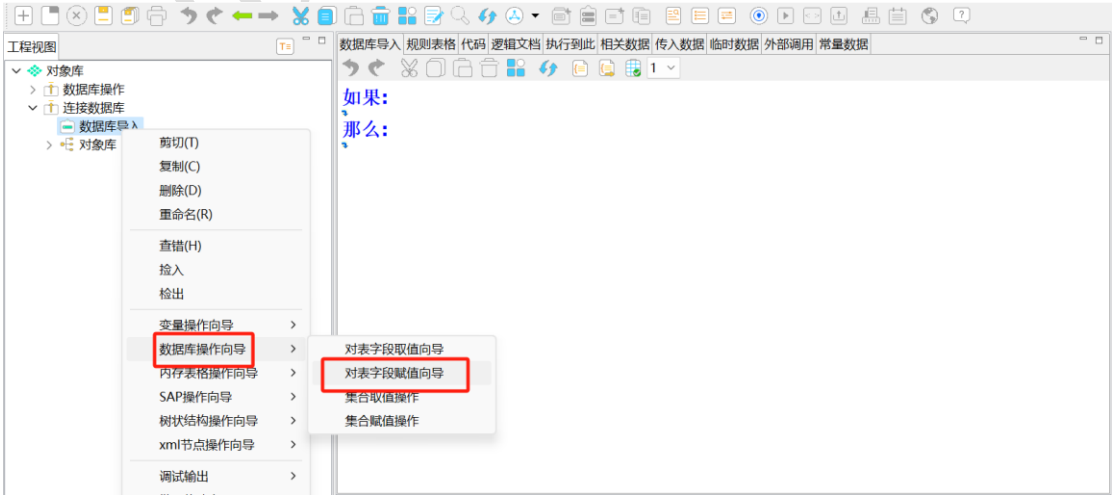
（1）数据库插入

点击“连接数据库”规则包，选择“添加规则”，并将其命名为“数据库插入”：



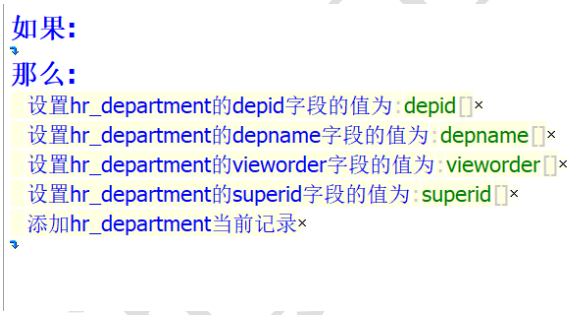
在上面的案例中，我们是通过在规则窗口中“添加动作”来给每个变量逐条赋值的，这样当数据较多时，添加效率就比较低了，这里采用自动匹配生成的方法。

右键“数据库插入”规则，在列表中选择“数据库操作向导”，这里我们是要往数据库中插入数据，因此选择“对表字段赋值向导”：



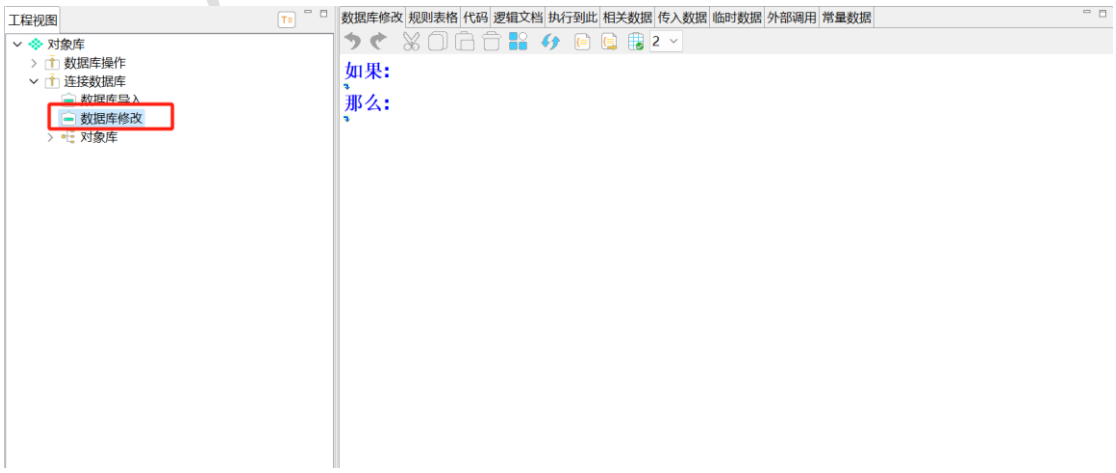
勾选需要赋值的字段，并选择“包含插入操作”：

点击确定，生成赋值插入操作，并根据显示名称在对象库“传入数据”中自动匹配。

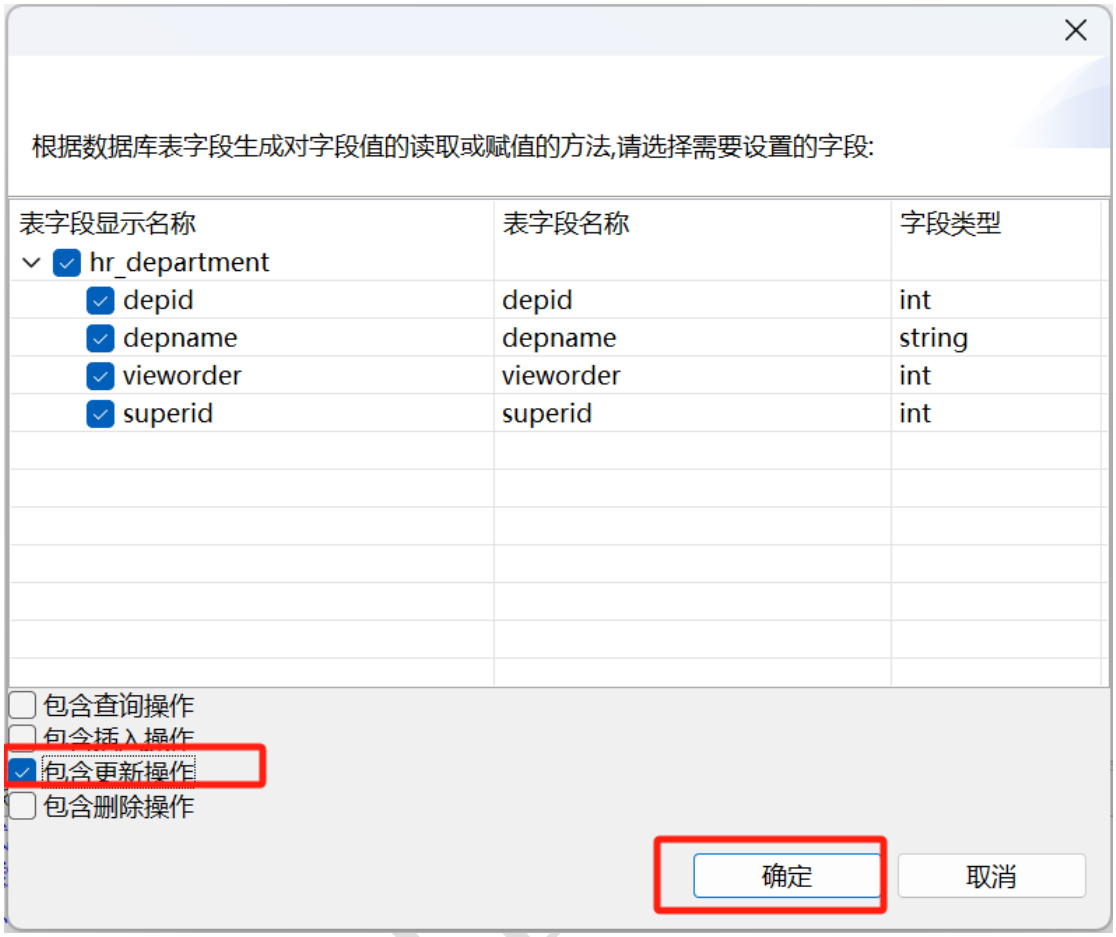


(2) 数据库修改

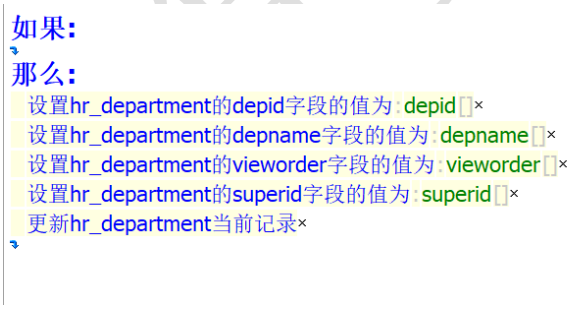
点击“连接数据库”规则包，选择“添加规则”，并将其命名为“数据库修改”：



右键“数据库修改”规则，在列表中选择“数据库操作向导”→“对表字段赋值向导”。勾选需要修改的字段，并选择“包含更新操作”：

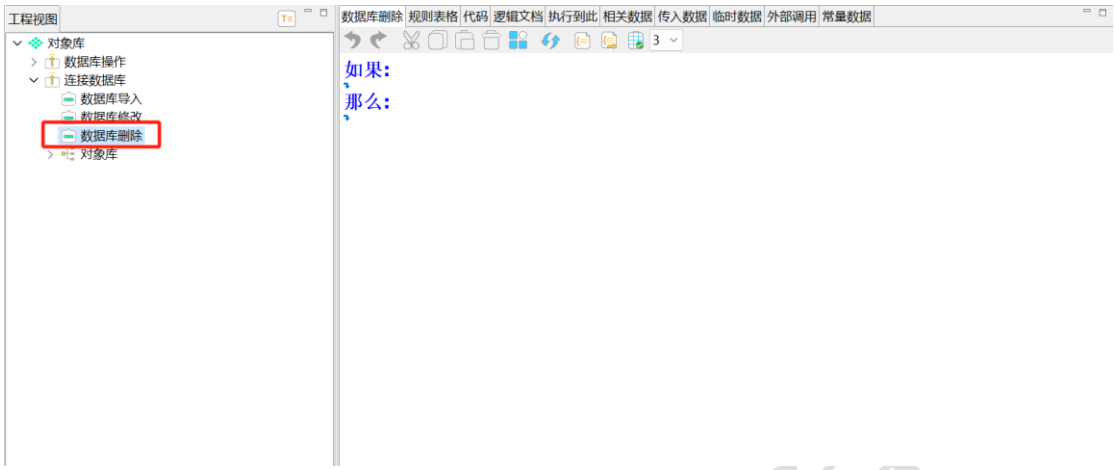


点击确定，生成更新操作，并根据显示名称在对象库“传入数据”中自动匹配。



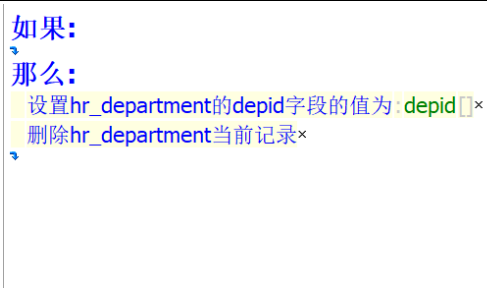
(3) 数据库删除

点击“连接数据库”规则包，选择“添加规则”，并将其命名为“数据库删除”：



右键“数据库删除”规则，在列表中选择“数据库操作向导”→“对表字段赋值向导”。勾选需要删除的记录的 ID，并选择“包含删除操作”：

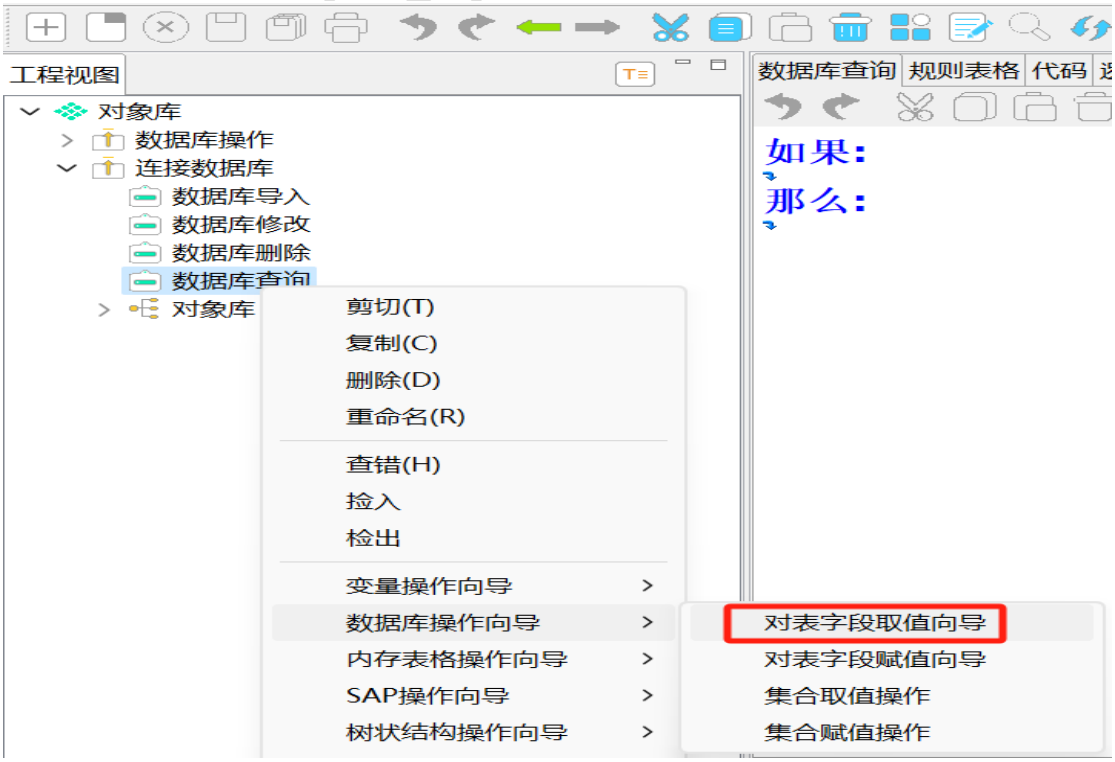
点击确定，根据 ID 生成删除操作：



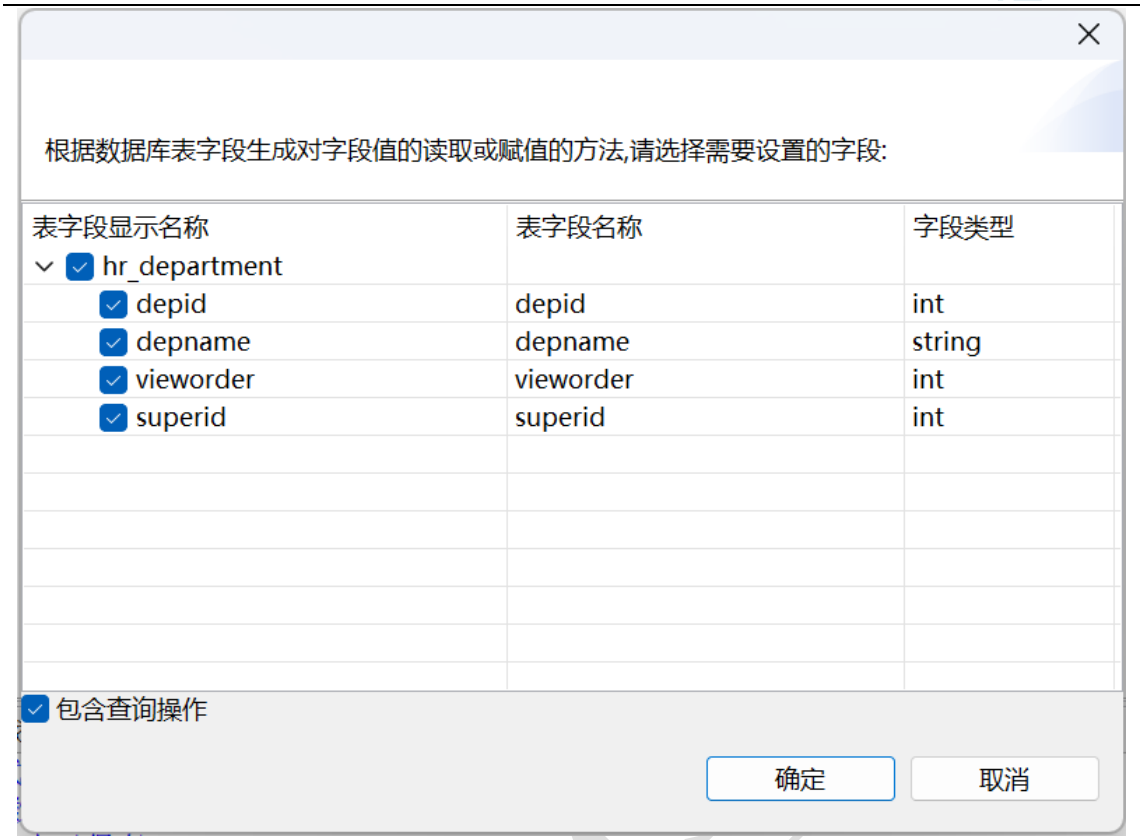
(4) 数据库查询

点击“连接数据库”规则包，选择“添加规则”，并将其命名为“数据库查询”：

右键“数据库查询”规则，在列表中选择“数据库操作向导”，这里我们要获取数据库中数据，因此选择“对表字段取值向导”：



弹出窗体，勾选“hr_department”及“包含查询操作”，最后点击确定，如下图：



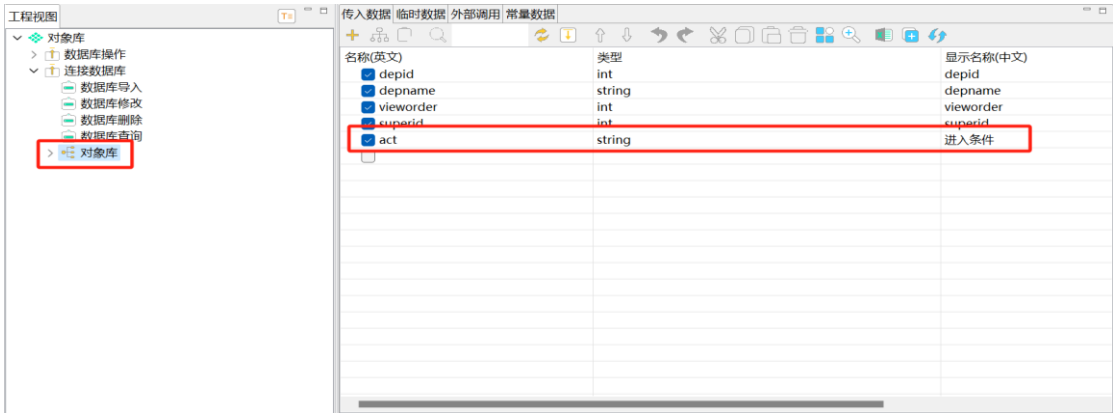
点击确定，规则自动生成，如下图：

这样“增加”、“修改”、“删除”、“查询”规则就配置完成了。

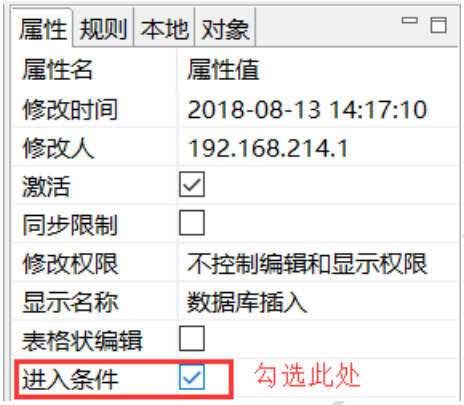
(5) 设置进入条件

由于在规则包“连接数据库”中有“增加”、“修改”、“删除”、“查询”等规则，而在实际测试中则是根据需求去调用不同的规则，因此在这里我们为每个规则都添加一个进入条件。

首先需要定义一个条件变量，点击“对象库”，在传入数据下的编辑窗体的“名称（英文）”列下双击添加变量：



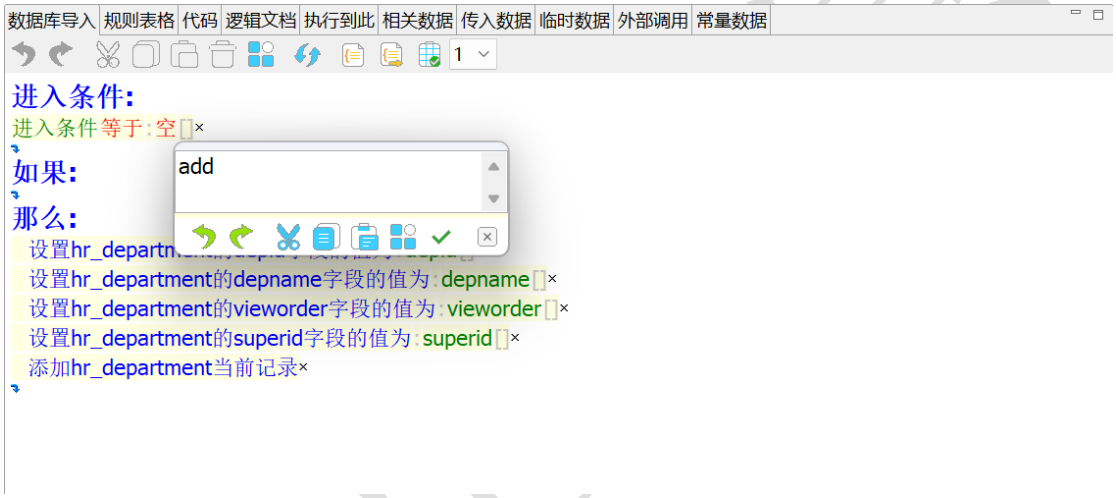
变量定义完成之后，开始修改规则，分别为规则“添加”“修改”“删除”“查询”添加进入条件，在属性窗口中，勾选属性“进入条件”设置为如下：



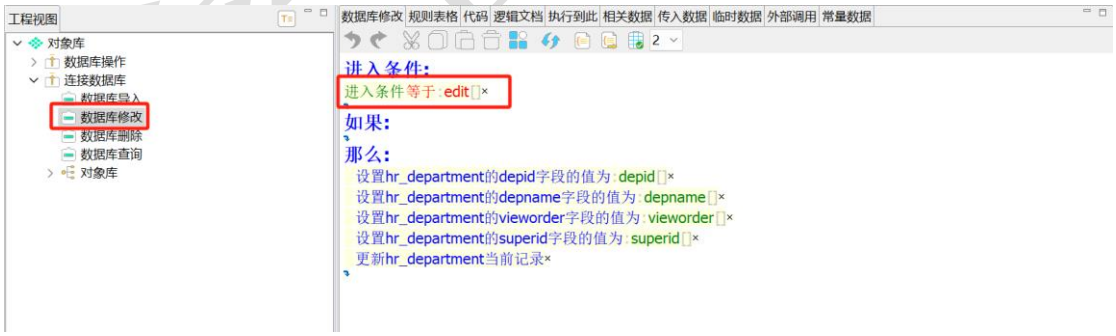
勾选之后在编辑窗口中会出现如下所示：

修改“数据库插入”规则，设置其进入条件，如下图

设置进入条件为“进入条件 等于 add”：



同上修改规则“数据库修改”、“数据库删除”、“数据库查询”的进入条件，完成后如下图所示：

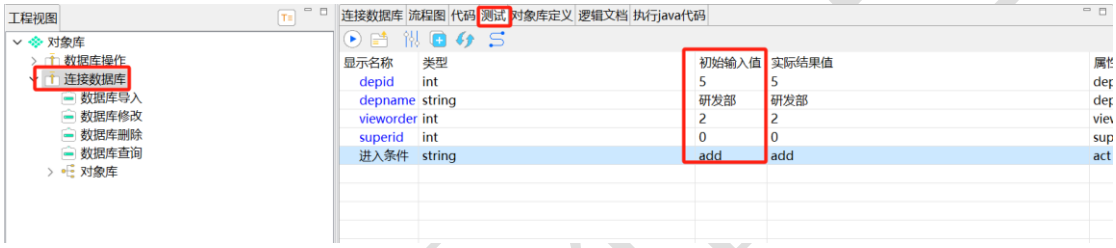


以上规则都完成之后，点击“ ”保存并编译。

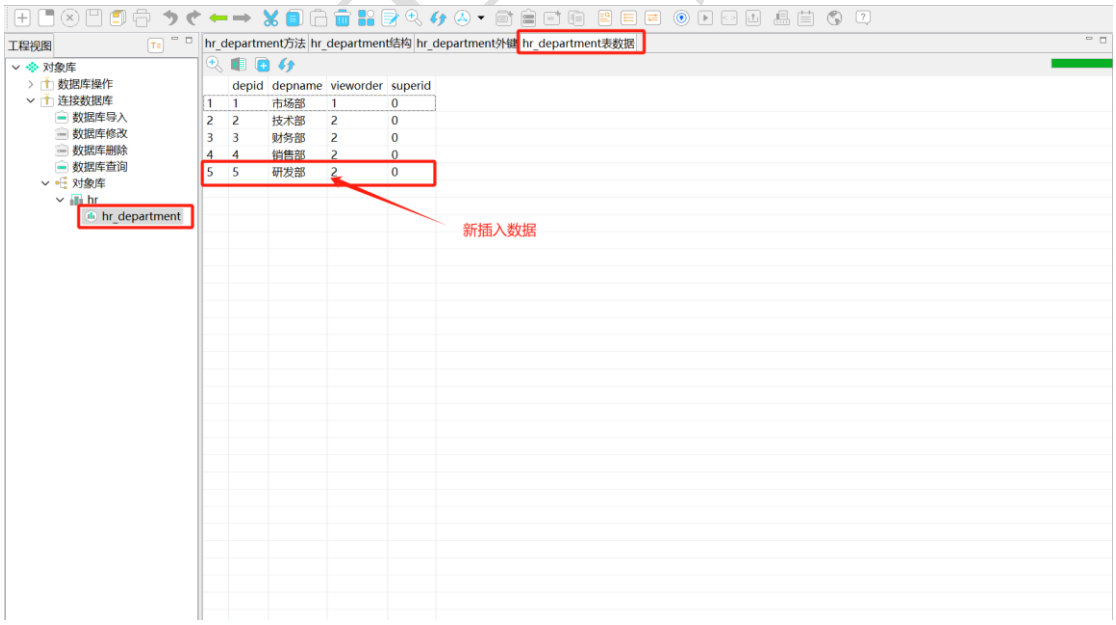
5．测试

点击“连接数据库”规则包，在测试界面中输入“初始输入值”进行测试：

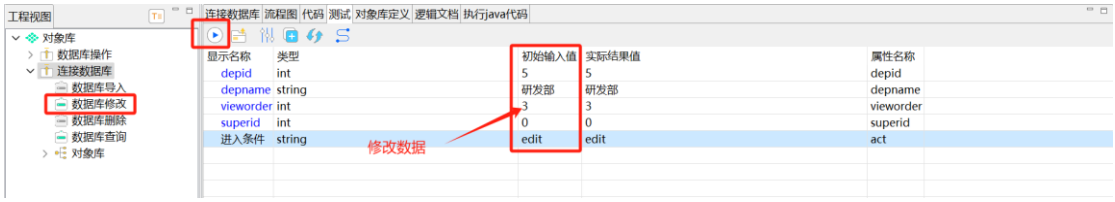
(1) 测试“数据库插入”规则：



点击对象库下“hr_department”在“hr_department 表数据”中查看插入数据。

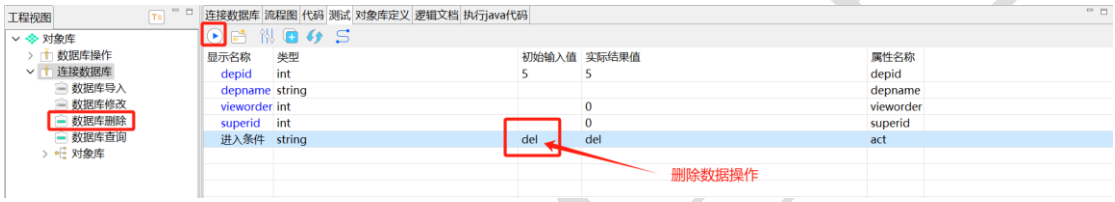


(2) 测试“数据库修改”规则：

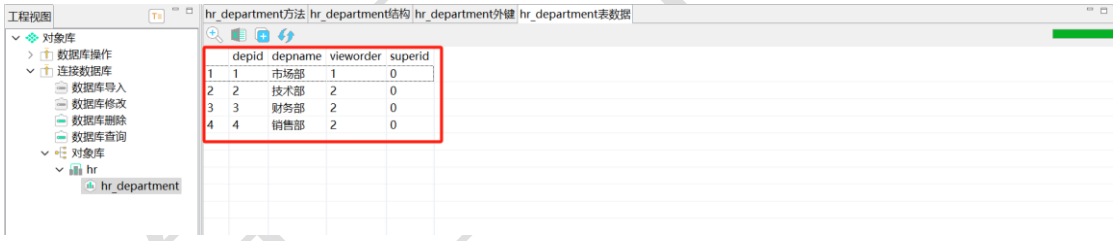


在数据库表中查看修改之后的数据：

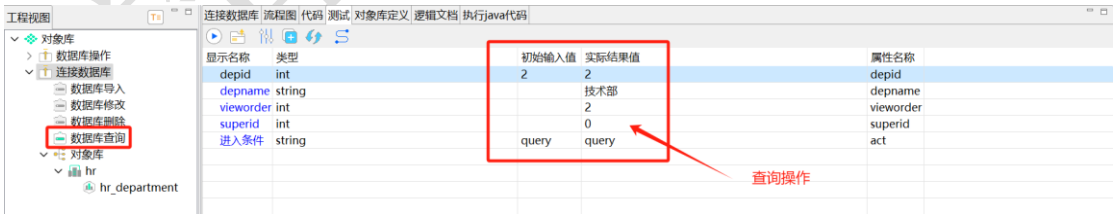
(3) 测试“数据库删除”规则：



查看表数据：

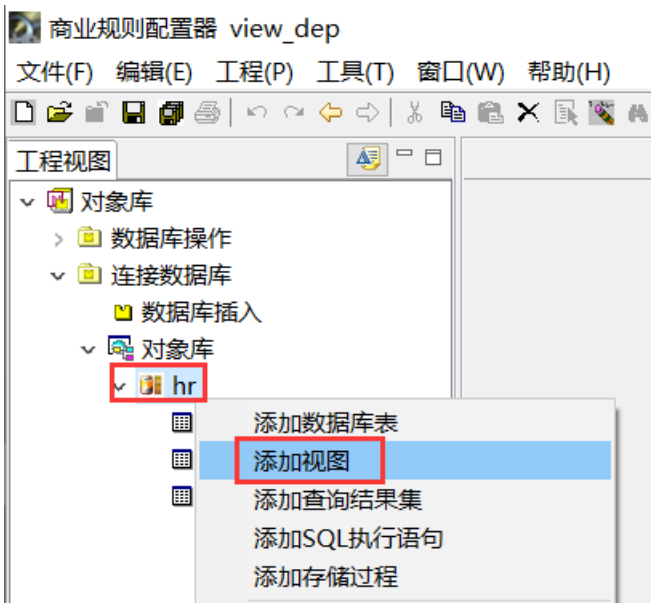


(4) 测试“数据库查询”规则：

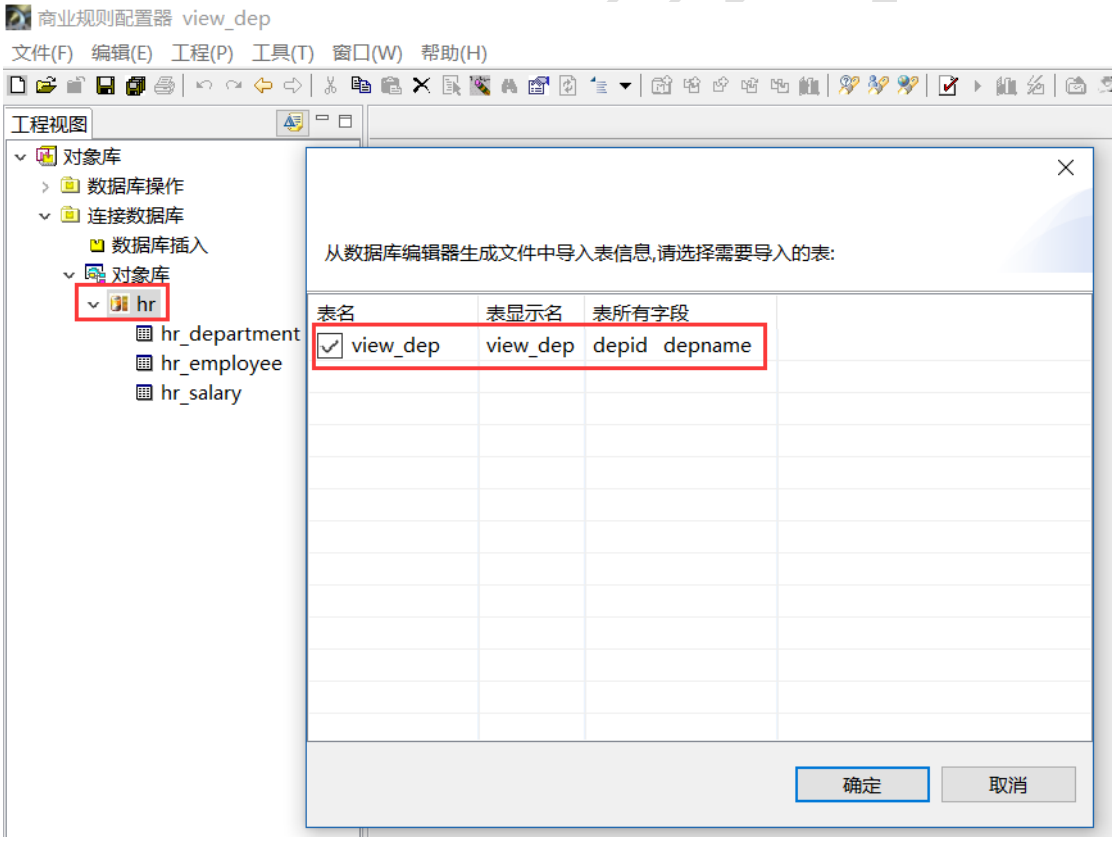


2.3.1.2. 视图

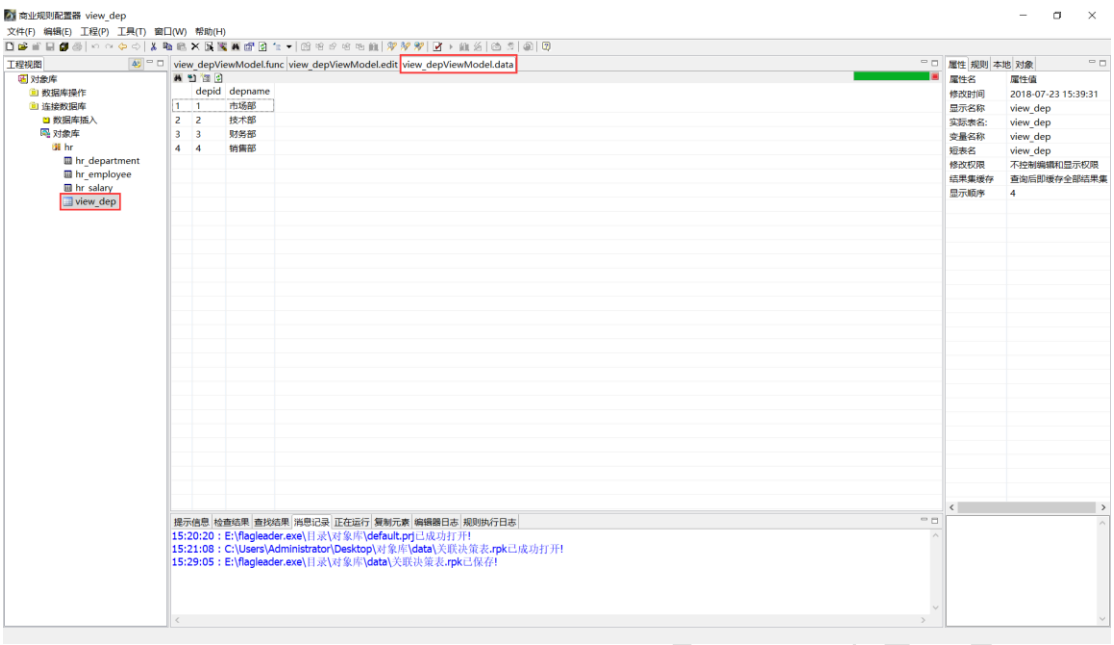
右键数据库“hr”选择“添加视图”：



勾选上视图“view_dep”，点击“确定”，如下图：

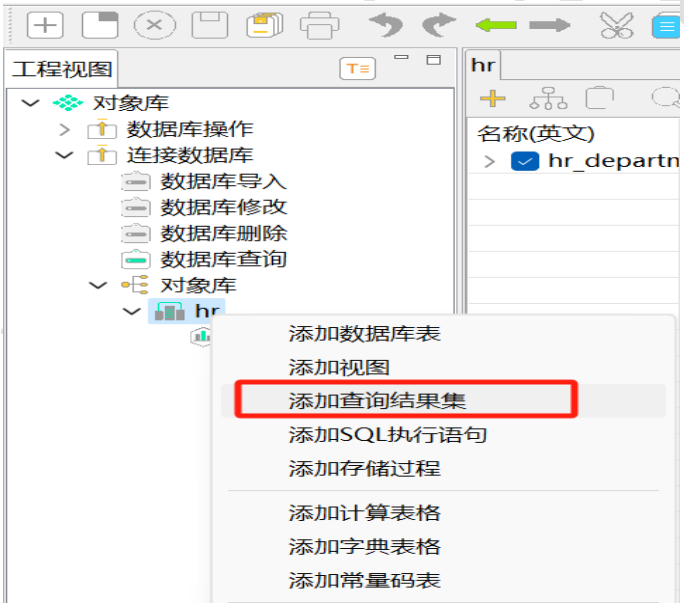


点击视图“view_dep”，选择选项卡“view_depViewModel.data”，查询视图信息，如下图：

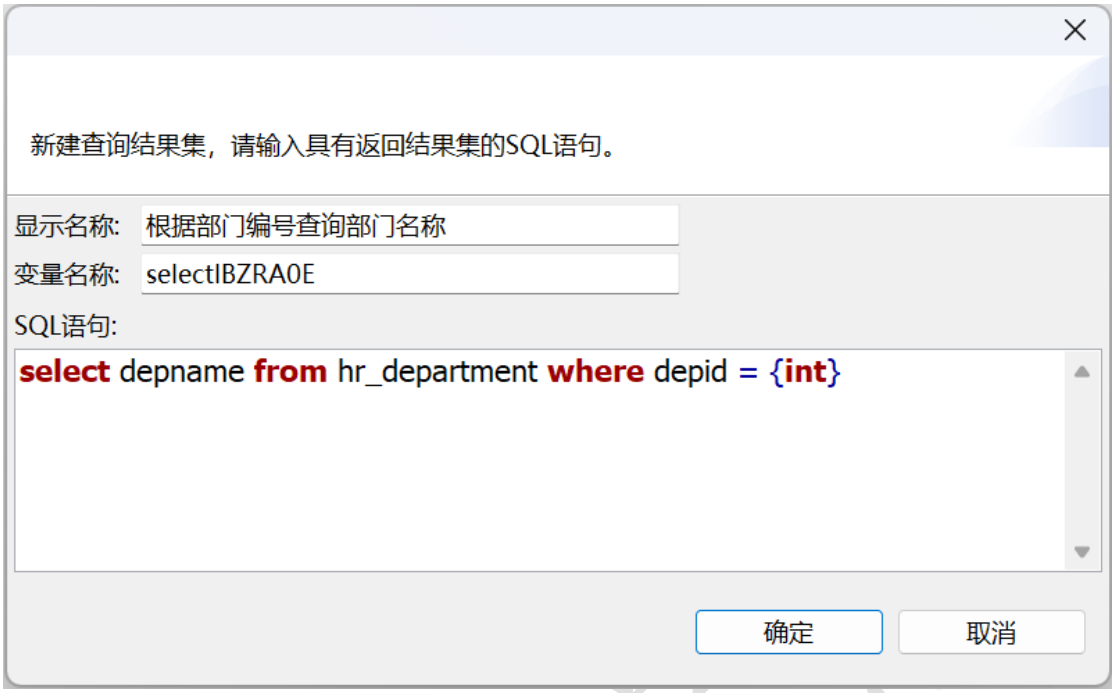


2.3.1.3. 查询结果集

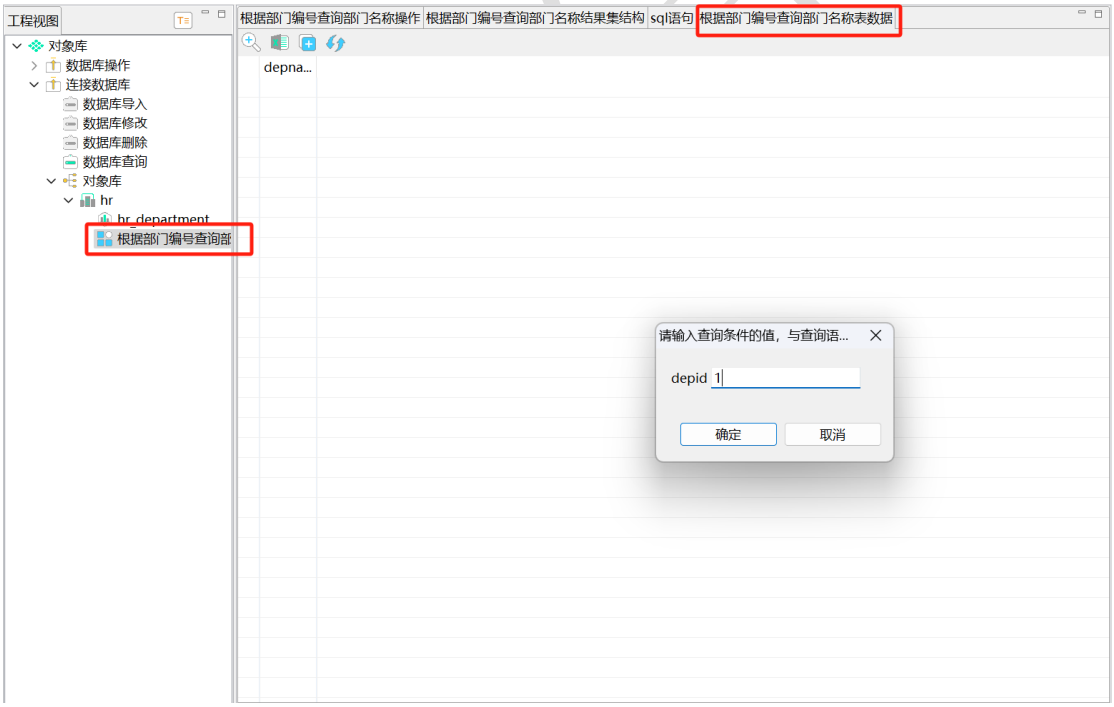
右键点击数据库连接文件“hr”，点击菜单项中的“添加查询结果集”，如下图：



在弹出窗口中将显示名称改为“根据部门编号查询部门名称”，再编写 sql 语句，如下图：



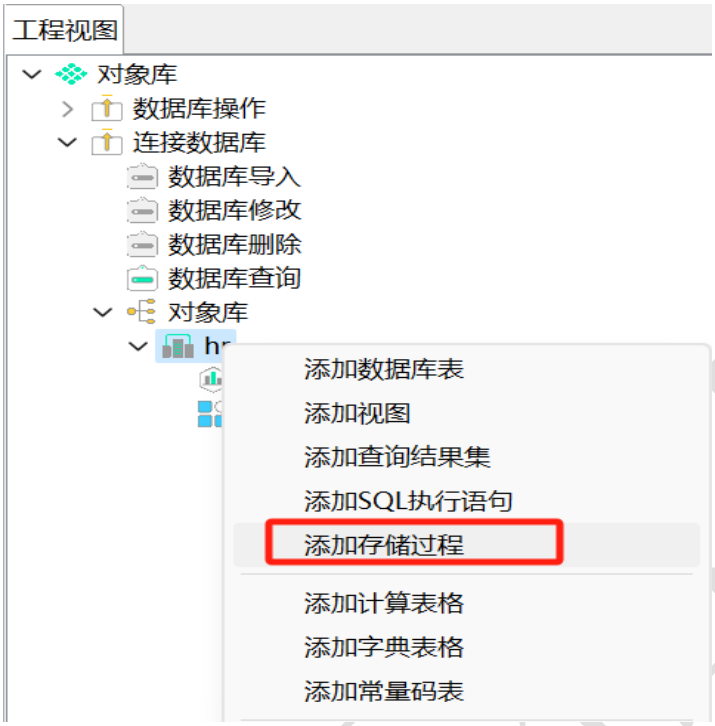
点击确定后，查询结果集就生成了，查看“根据部门编号查询部门名称”结果集，点击“根据部门编号查询部门名称表数据”在弹出窗体中的“dipid”文本框中输入“1”，如下图：



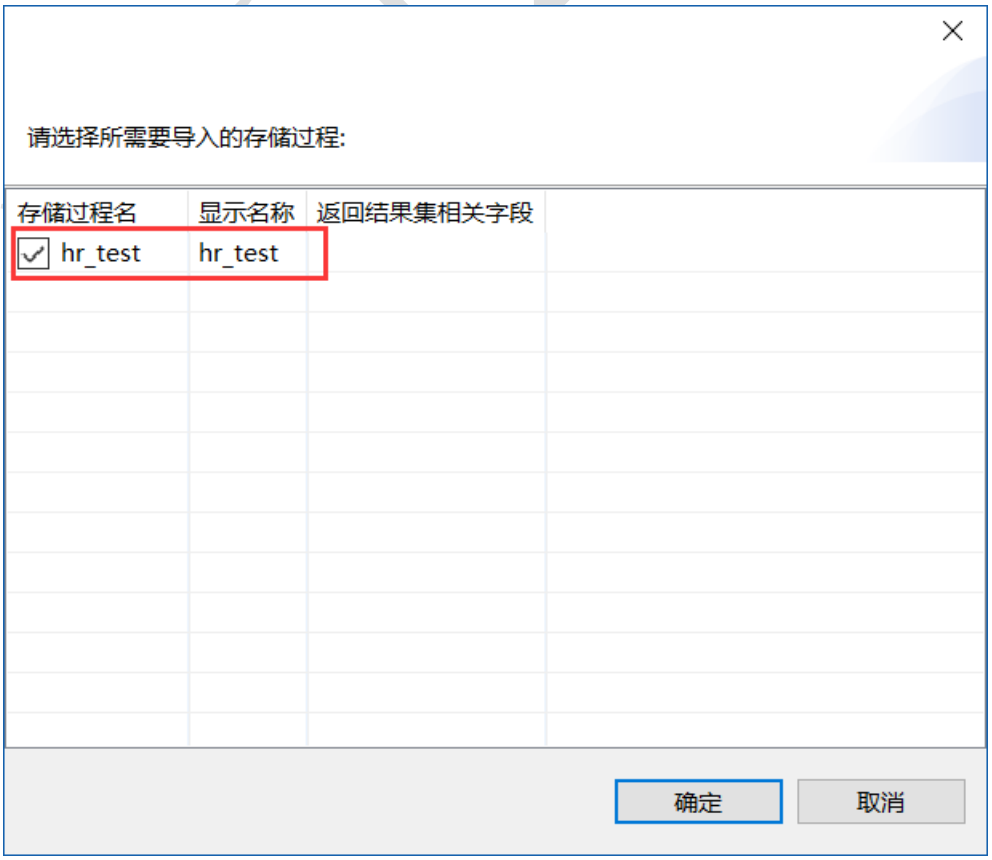
点击确定，显示如下：

2.3.1.4. 存储过程

右键点击数据库连接文件“hr”，点击菜单项中的“添加存储过程”，如下图：

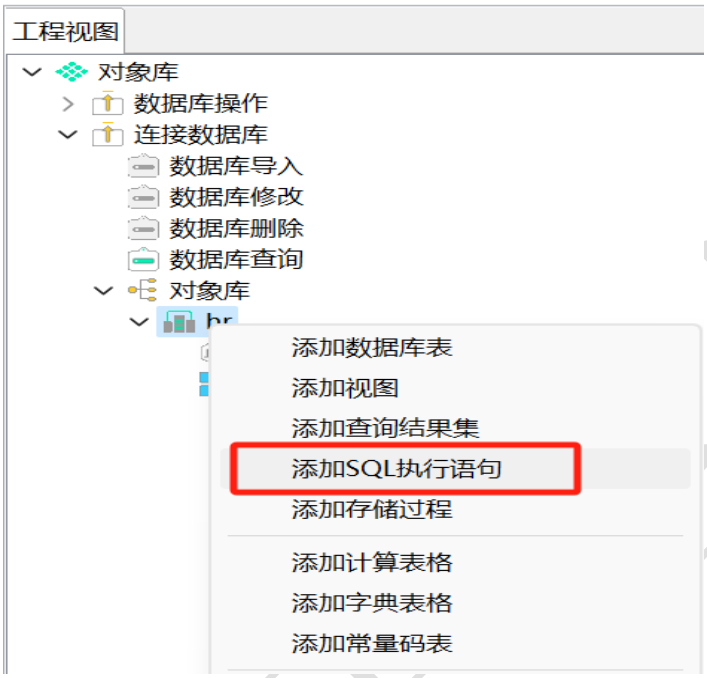


勾选上存储过程“hr_test”,点击“确定”，如下图：

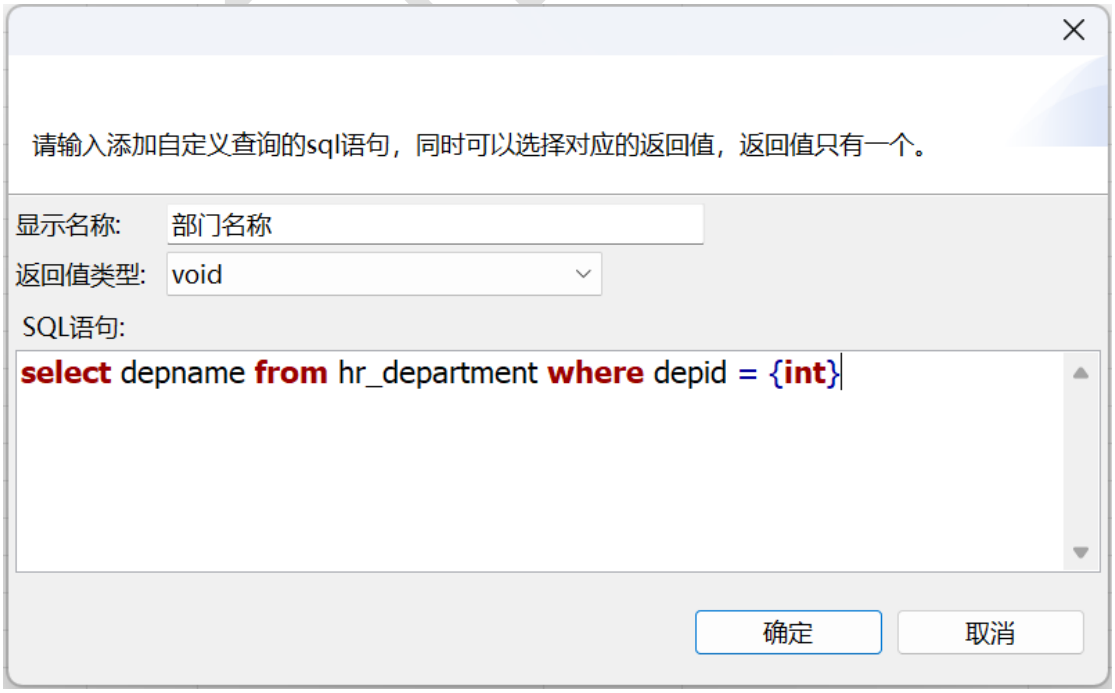


2.3.1.5. SQL 执行语句

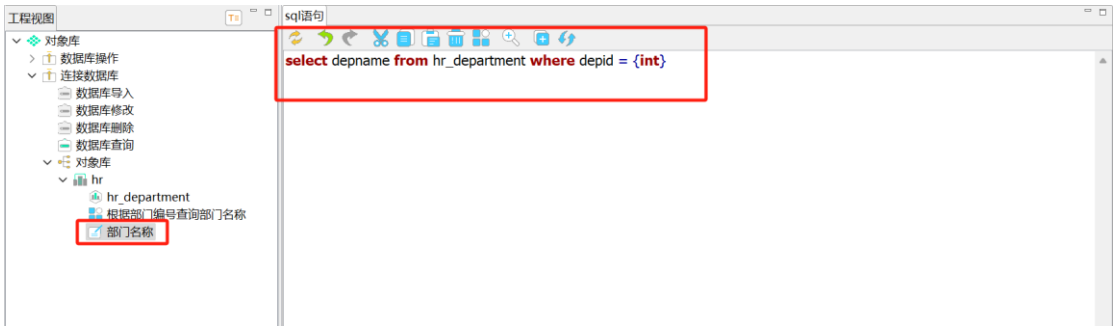
右键点击数据库连接文件“hr”，点击菜单项中的“添加 SQL 执行语句”，如下图：



弹出窗体中将显示名称改为“部门名称”，返回至类型设置为“string”，再编写 sql 语句，如下图：

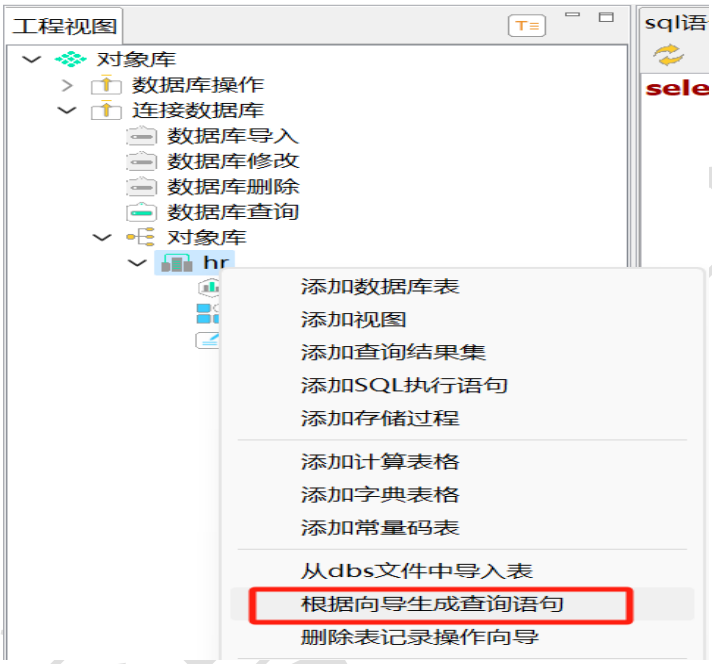


点击确定后，如下图：

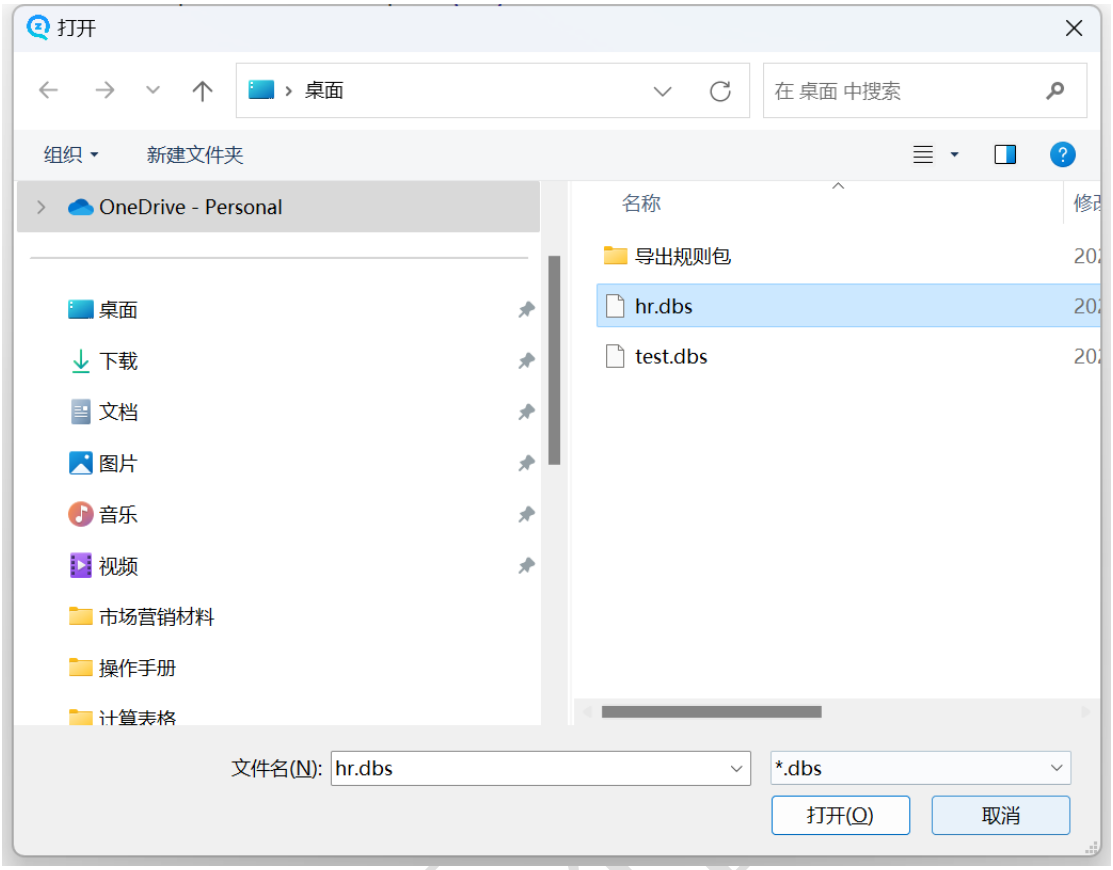


2.3.1.6. 根据向导生成查询语句

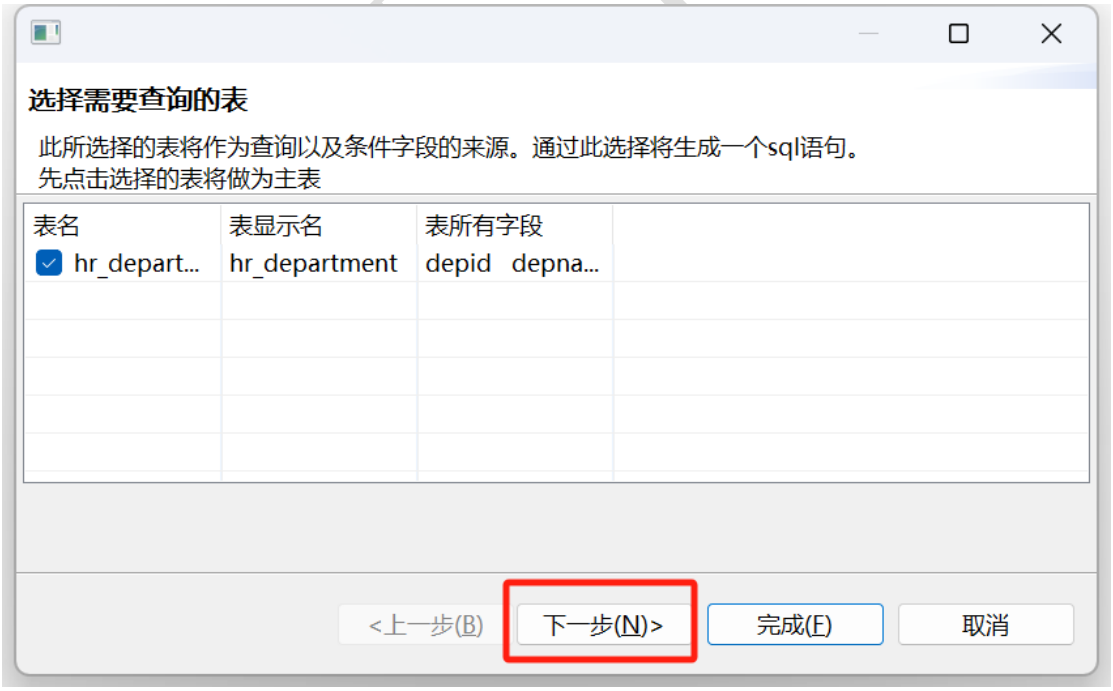
右键点击数据库连接文件“hr”，点击菜单项中的“根据向导生成查询语句”，如下图：



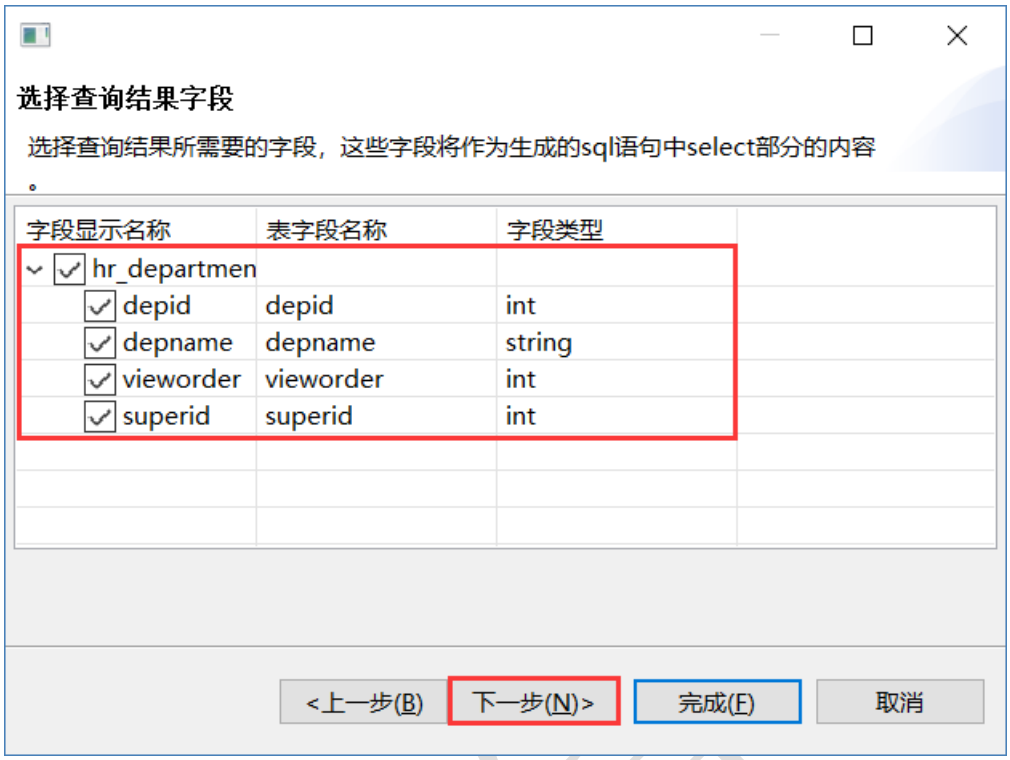
弹出“打开”窗体，开始寻找 dbs 文件，找到后点击打开，如下图：



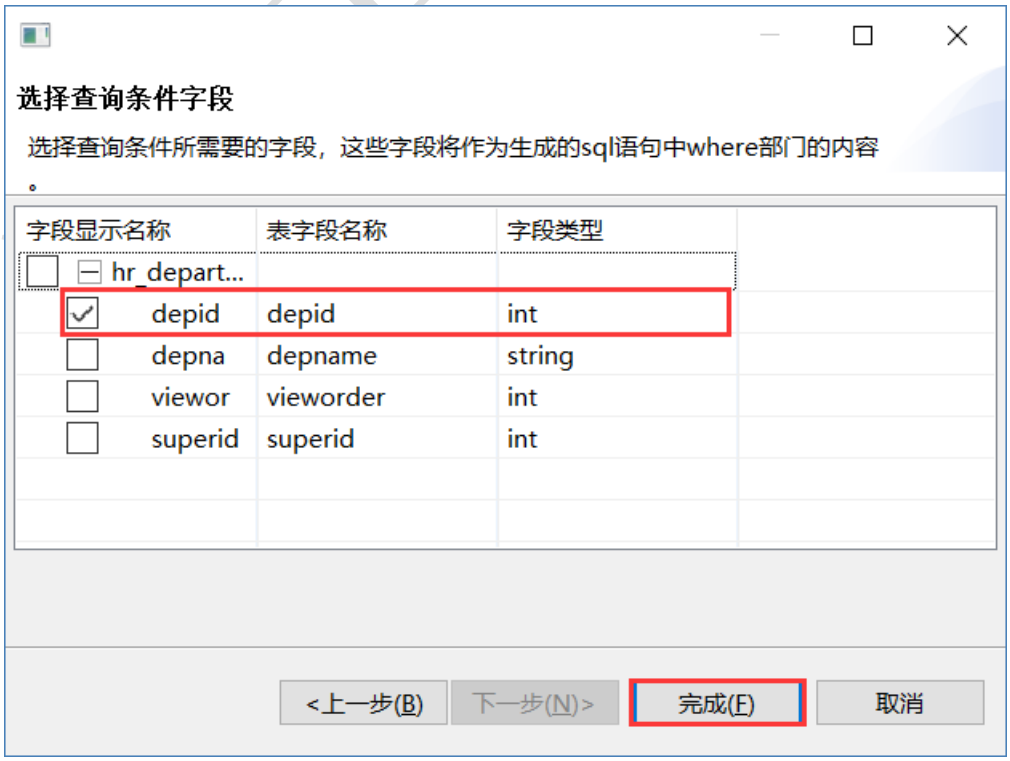
弹出窗体，选择需要查询的表，点击“下一步”，如下图：



选择了需要查询的表之后，在弹出窗体中，勾选需要查询的字段：

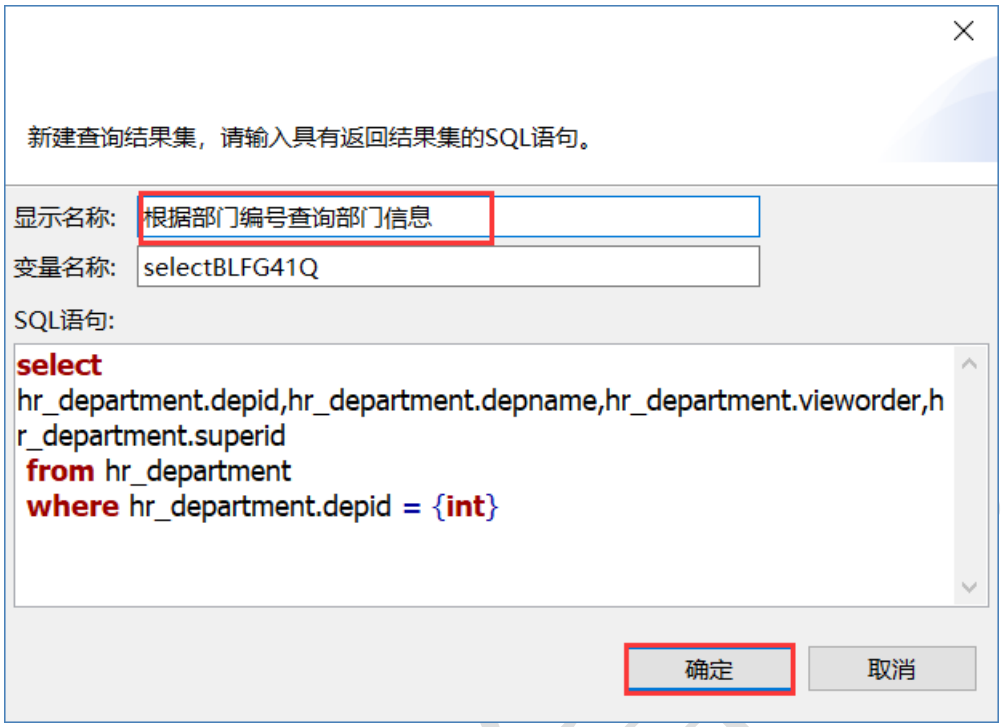


选择查询条件，如果不需要条件，可直接点击“完成”，如下图：

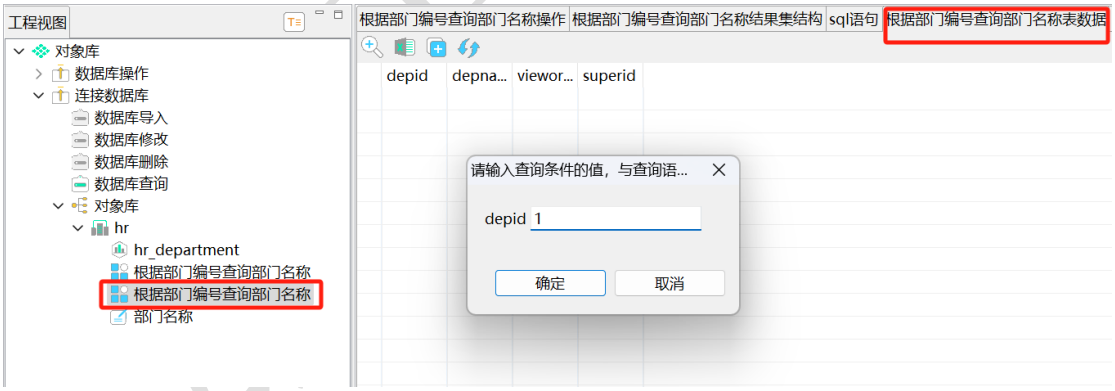


这样根据部门编号查询部门信息的查询语句就生成好了，将弹出窗口中的显示名称改为

“根据部门编号查询部门信息”，然后点击“确定”，如下图：



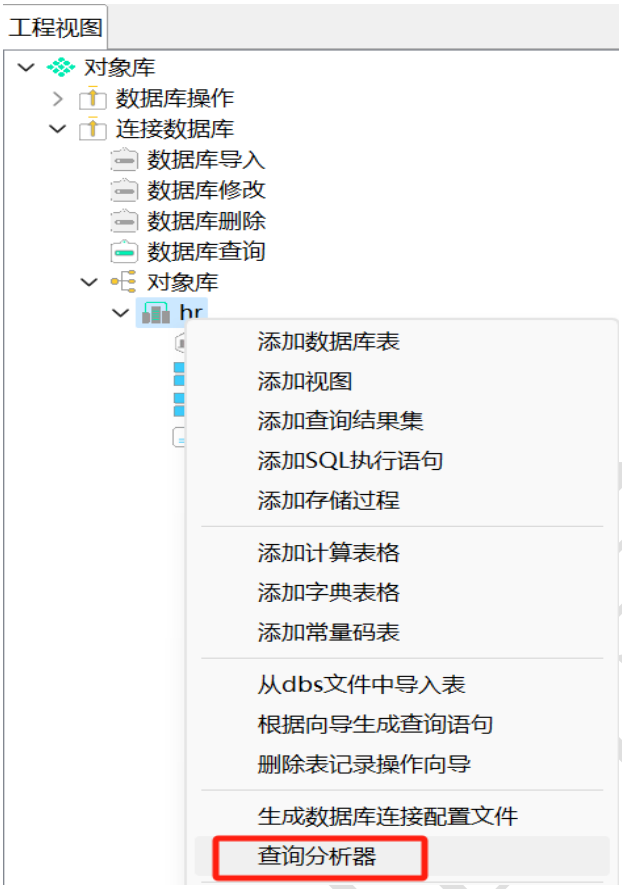
点击确定后，查询结果集也生成了,点击“根据部门编号查询部门信息”，查看“根据部门编号查询部门信息表数据”菜单项，会弹出窗体，该窗体的作用是输入条件，在 depid 对应的文本框中输入 1，显示如下：



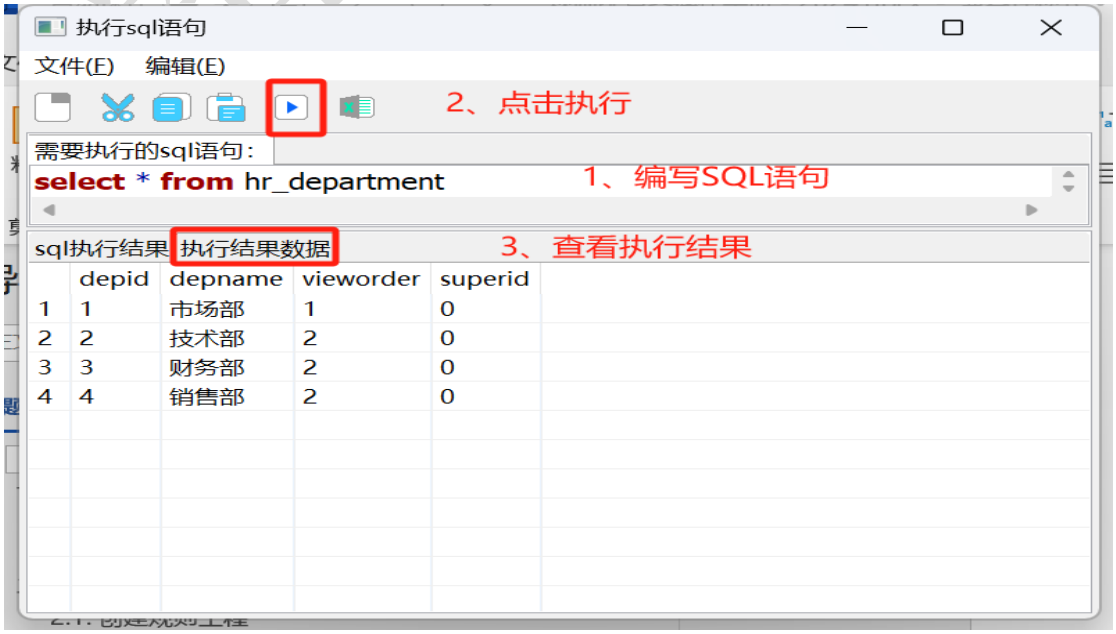
点击确定，查询结果如下：

2.3.1.7. 查询分析器

右键点击数据库连接文件“hr”，点击菜单项中的“查询分析器”，如下图：

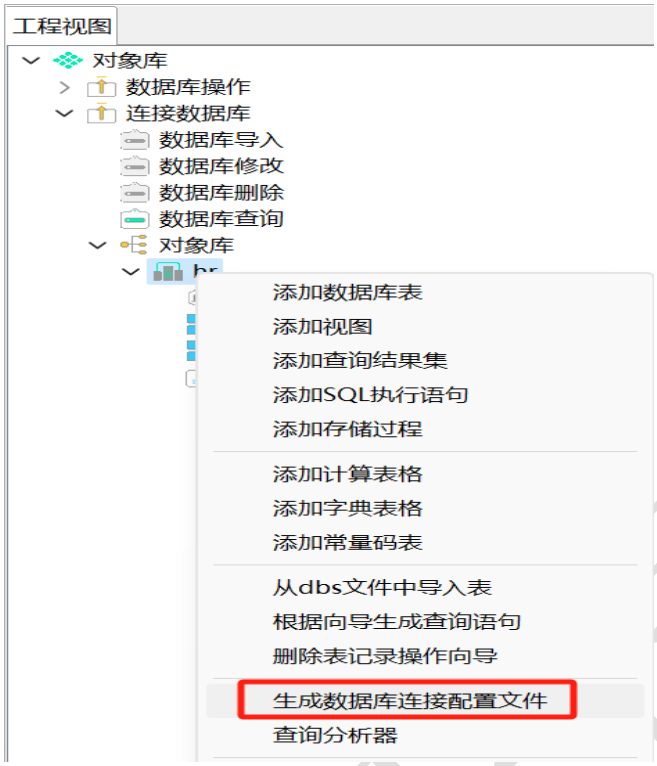


弹出窗体“执行 sql 语句”，在下面编辑 sql 语句，编辑完后点击  执行，在“执行结果数据”选项卡中查看执行结果，如下图：

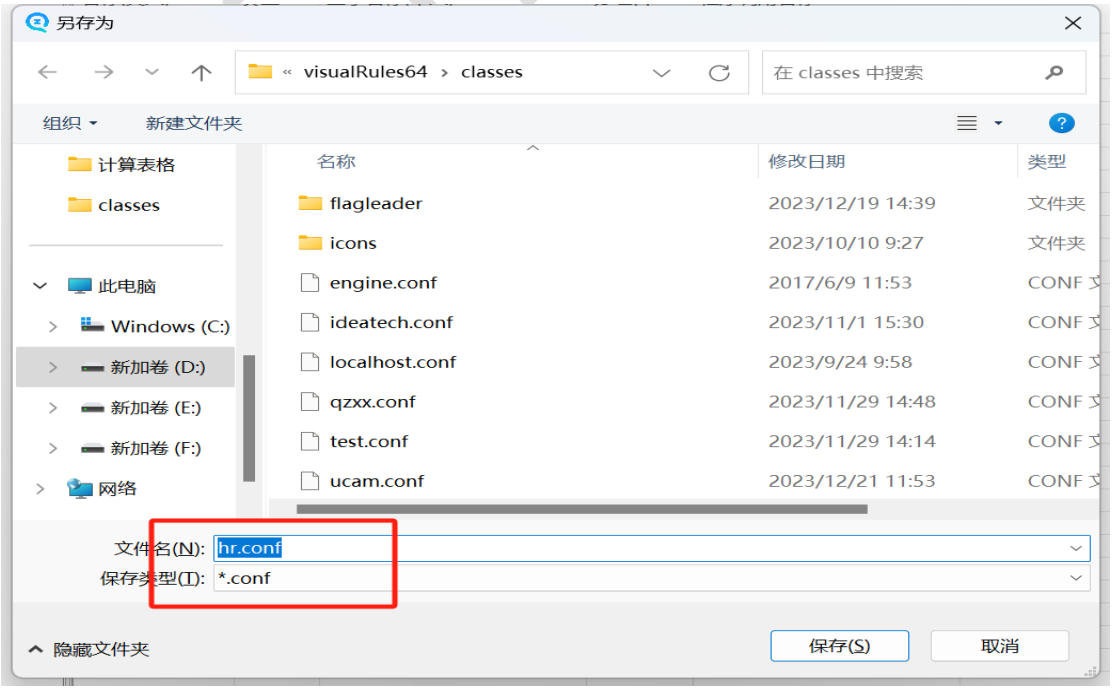


2.3.1.8. 生成数据库连接配置文件

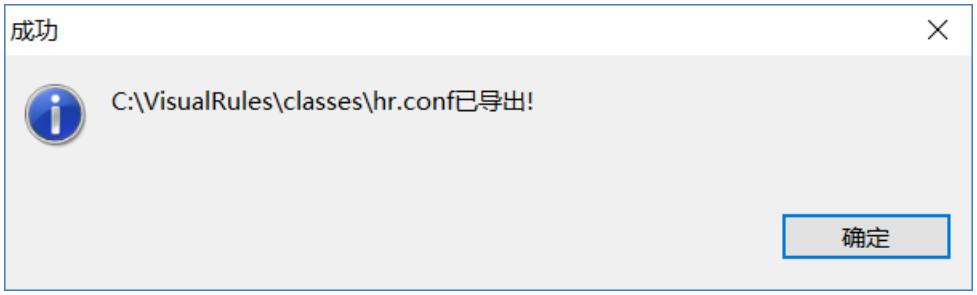
右键点击数据库连接文件“hr”，点击菜单项中的“生成数据库连接配置文件”，如下图：



弹出窗体，将生成的 hr.conf 文件放在规则引擎安装目录下的 class 目录下，点击保存，如下图：



保存后会弹出提示，提示如下：



操作完成。

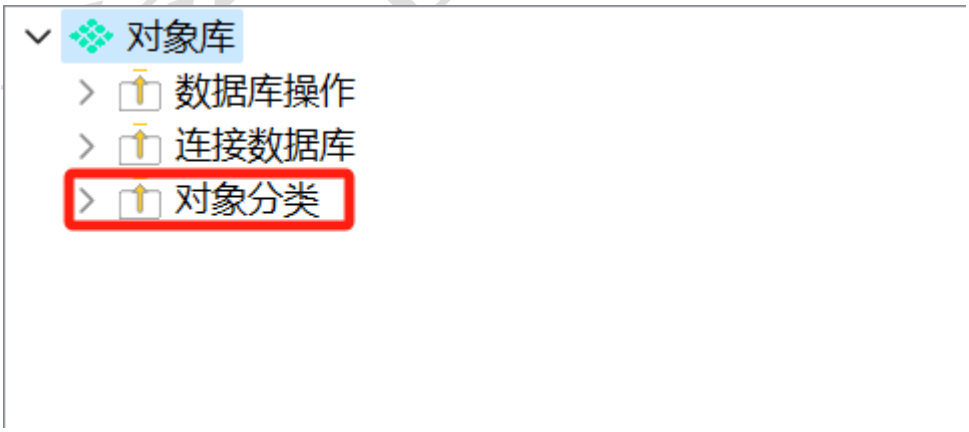
2.3.2. 对象分类

在“对象分类”中，我们可以添加“计算表格”、“常量码表”以及“数据字典”。下面通过对这三者在规则中的运用，来介绍“对象分类”的使用情况。

1．创建规则包

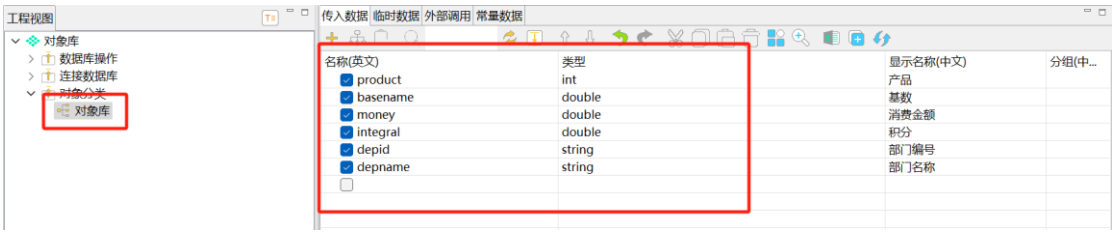
在“对象库”工程下创建规则包。点击“对象库”右键“新建规则包”：

将其命名为“对象分类”：



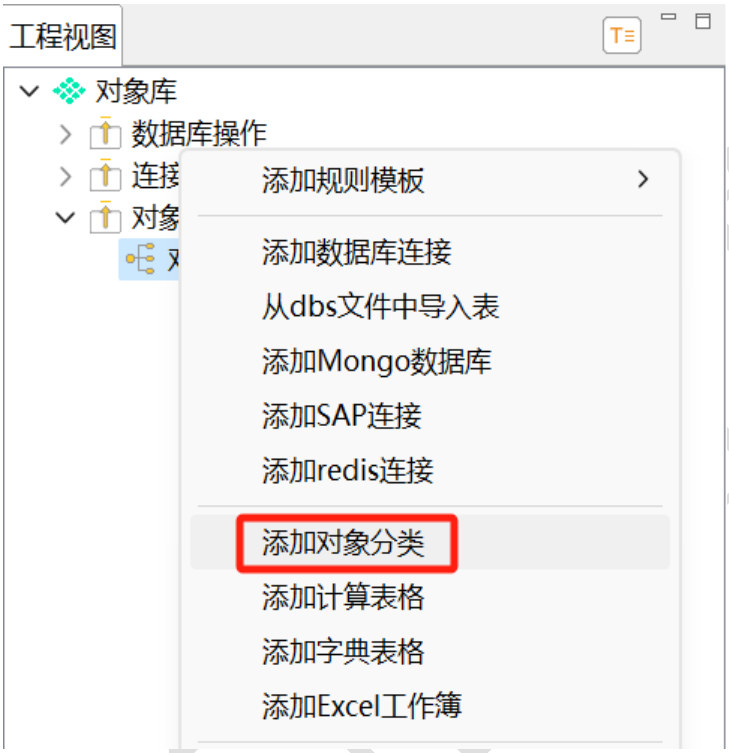
2．定义对象库变量

打开“对象分类”规则包，在对象库的传入数据中添加变量。依次添加“产品（product）、基数（basename）、消费金额（money）、积分（integral）、部门编号（depid）、部门名称（depname）”。



3．添加对象分类

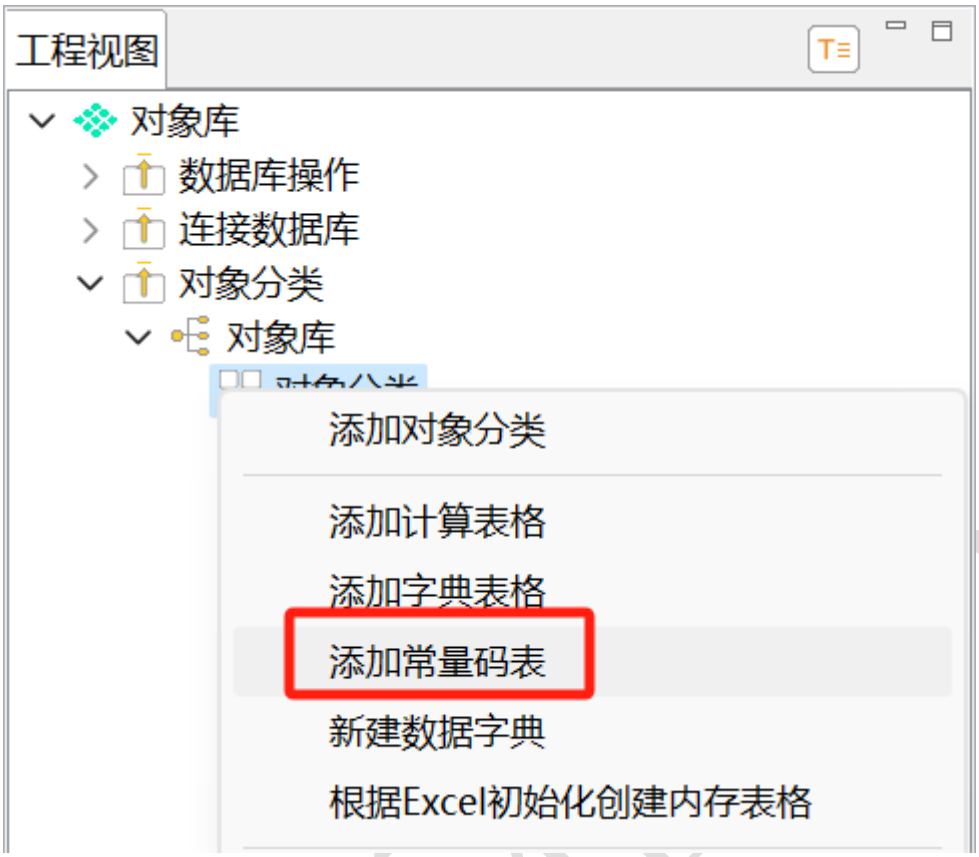
选择对象库，右键添加对象分类：

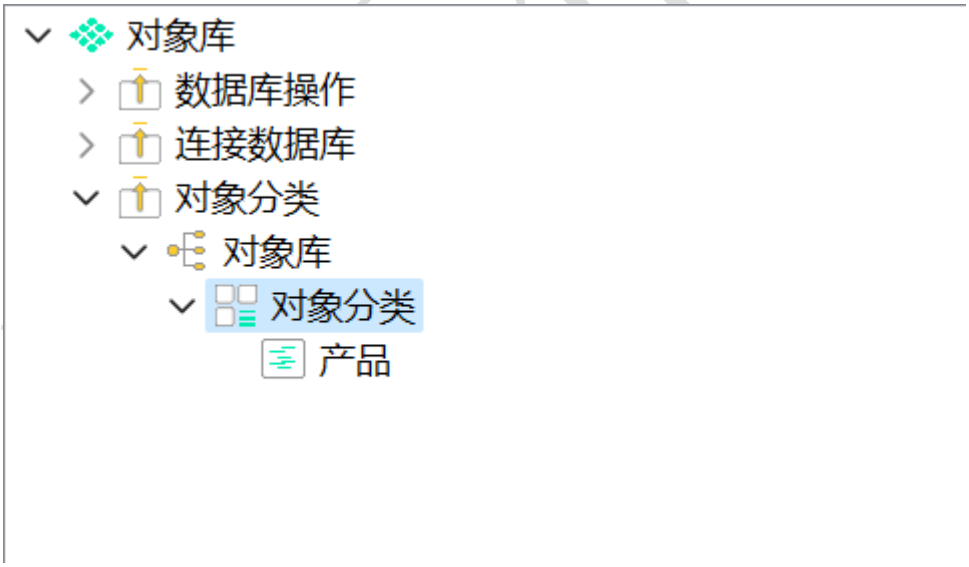


在对象分类下分别添加常量码表、计算表格以及数据字典。

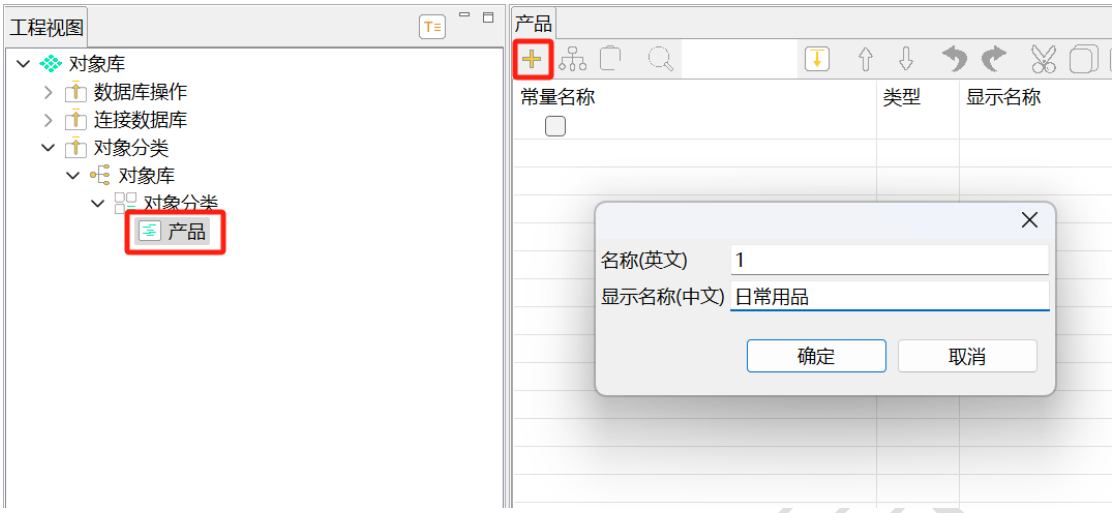
(1) 常量码表

右键对象分类，选择添加常量码表，在弹出框中输入显示名称为“产品”：

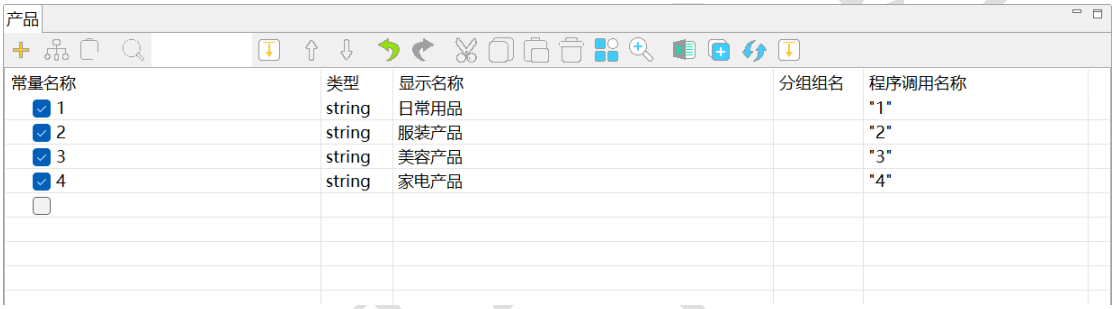




选择常量码表“产品”，在中间编辑窗体中点击“➕”添加常量：

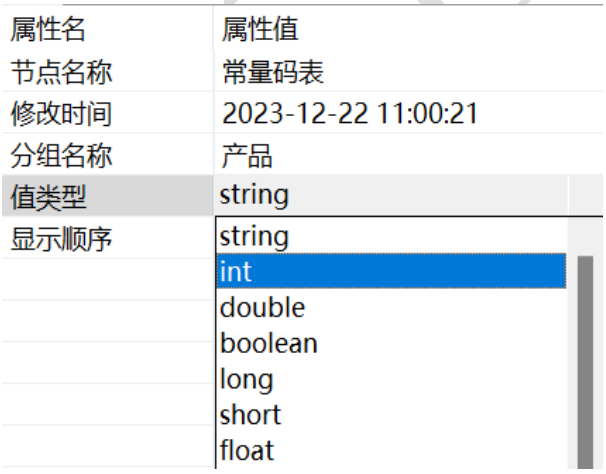


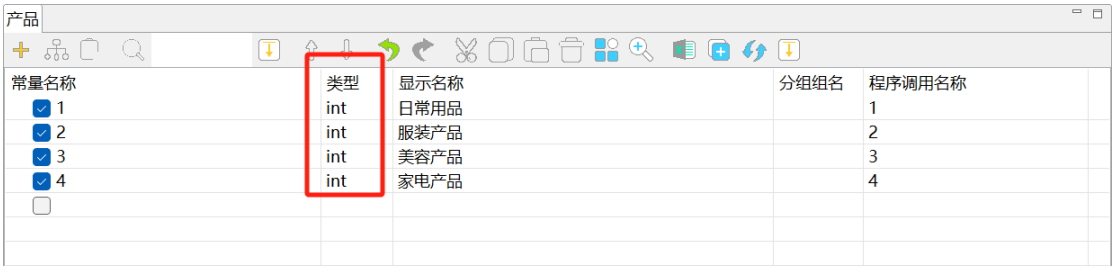
根据上述步骤再次添加常量“服装产品”、“美容产品”、“家电产品”：



由于这里的值类型默认的是 String，我们需要将其改为 int。在编辑窗体中将值类型由默认的 String 点击下拉更改为 int：

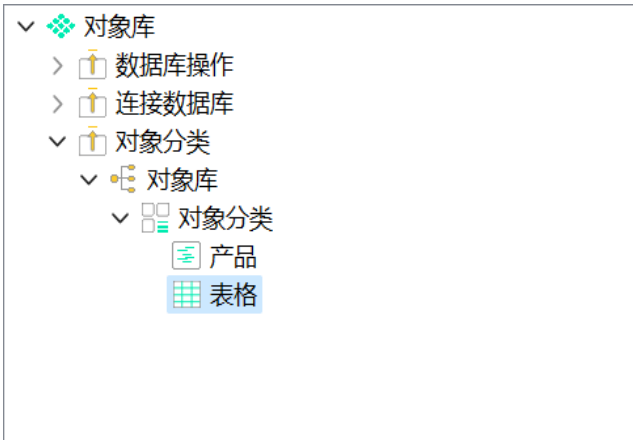
更改之后所有常量的值类型均为 int，如下图所示：





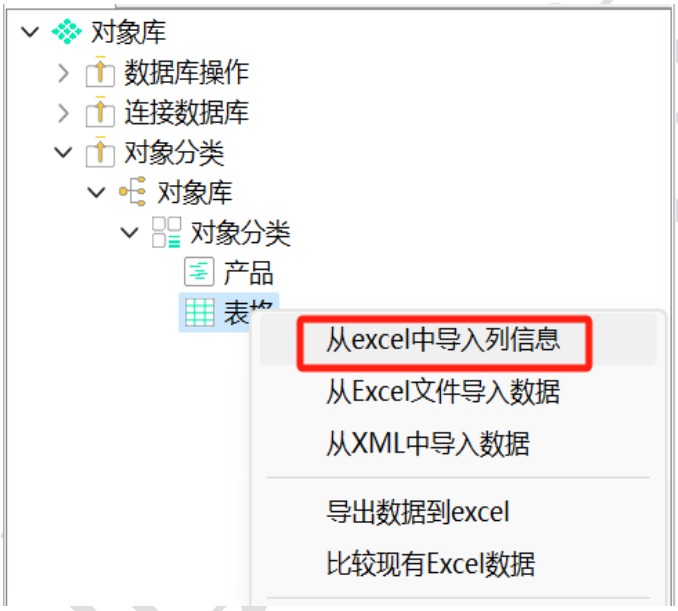
(2) 计算表格

选中对象分类，右键添加计算表格：

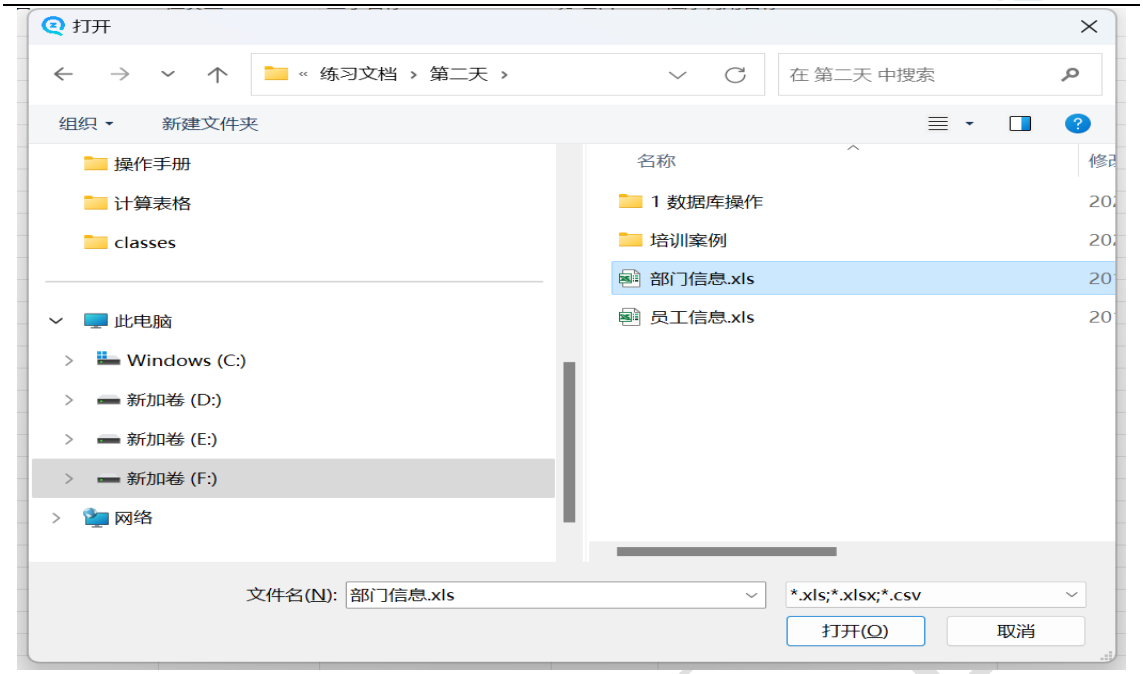


计算表格创建完成之后，需要为其添加数据，这里我们选择从 Excel 中导入。

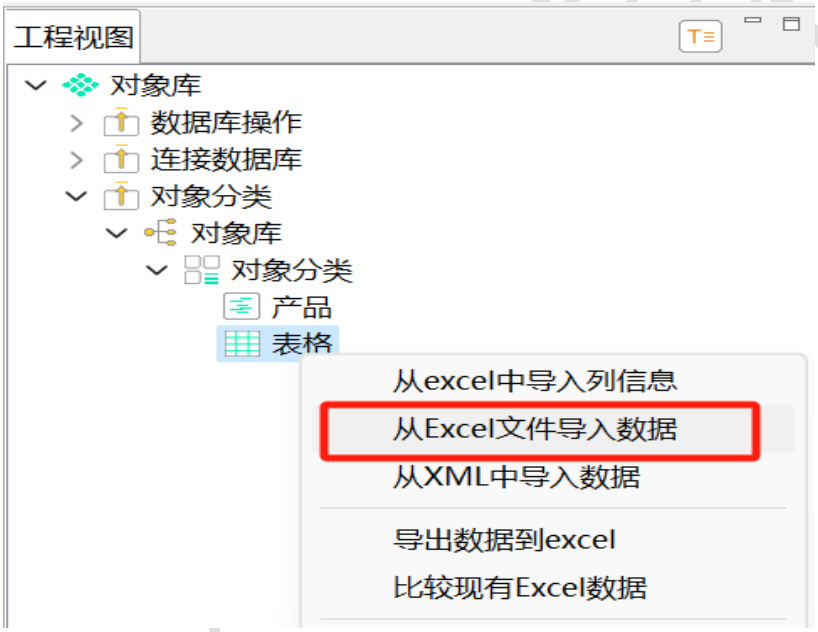
首先需要导入的是列信息。右键“表格”，选择“从 excel 中导入列信息”：



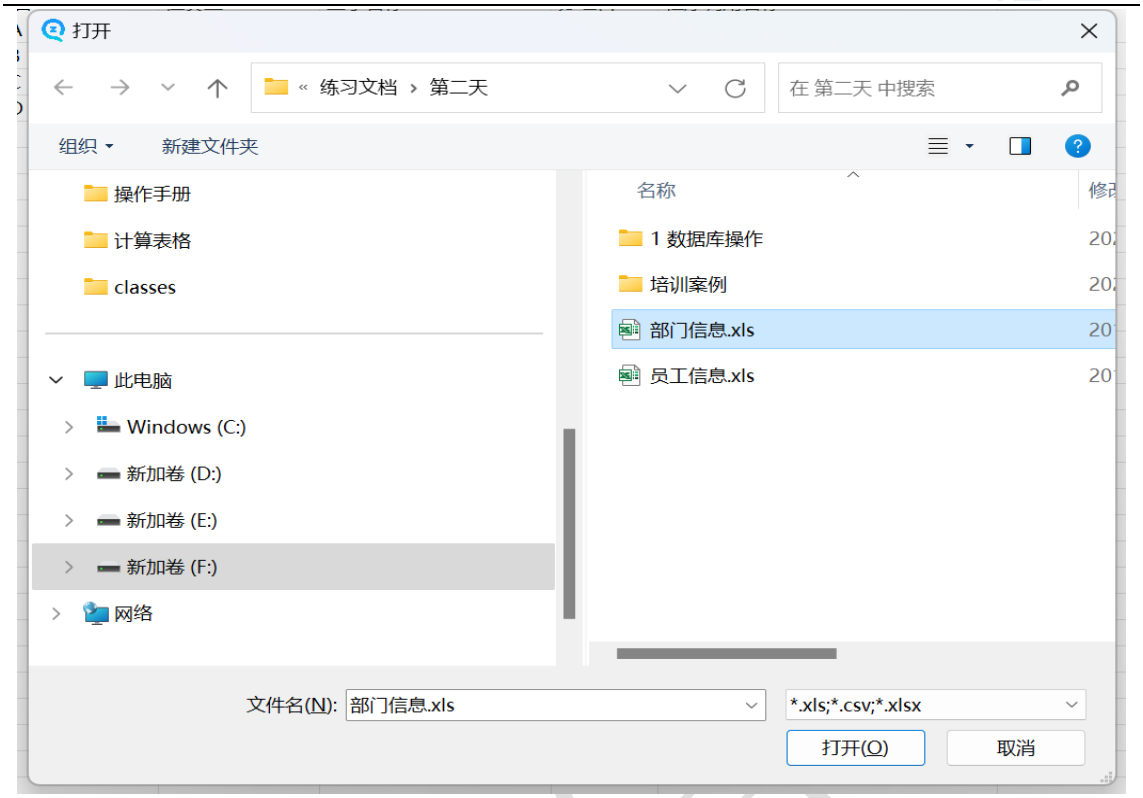
选择 xls 文件点击打开进行导入：



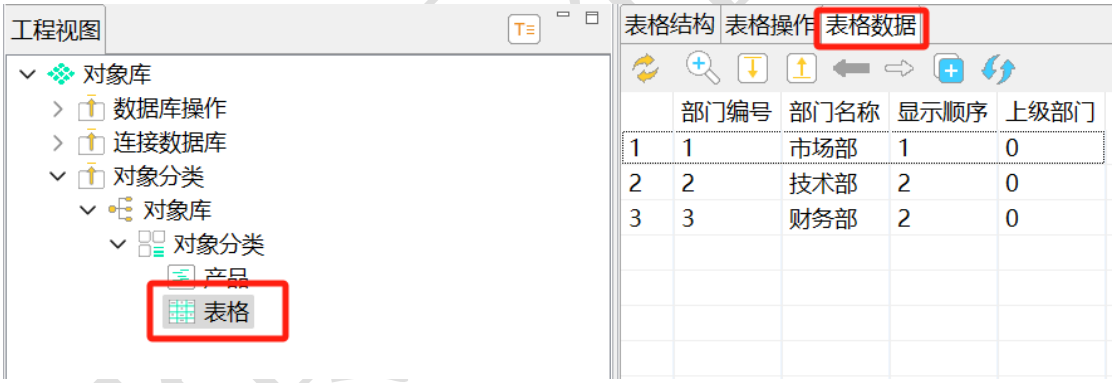
有了列之后，就要对表格添加数据。右键“表格”，选择“从 Excel 文件中导入数据”：



在弹出窗口中继续选择“部门信息.xls”：

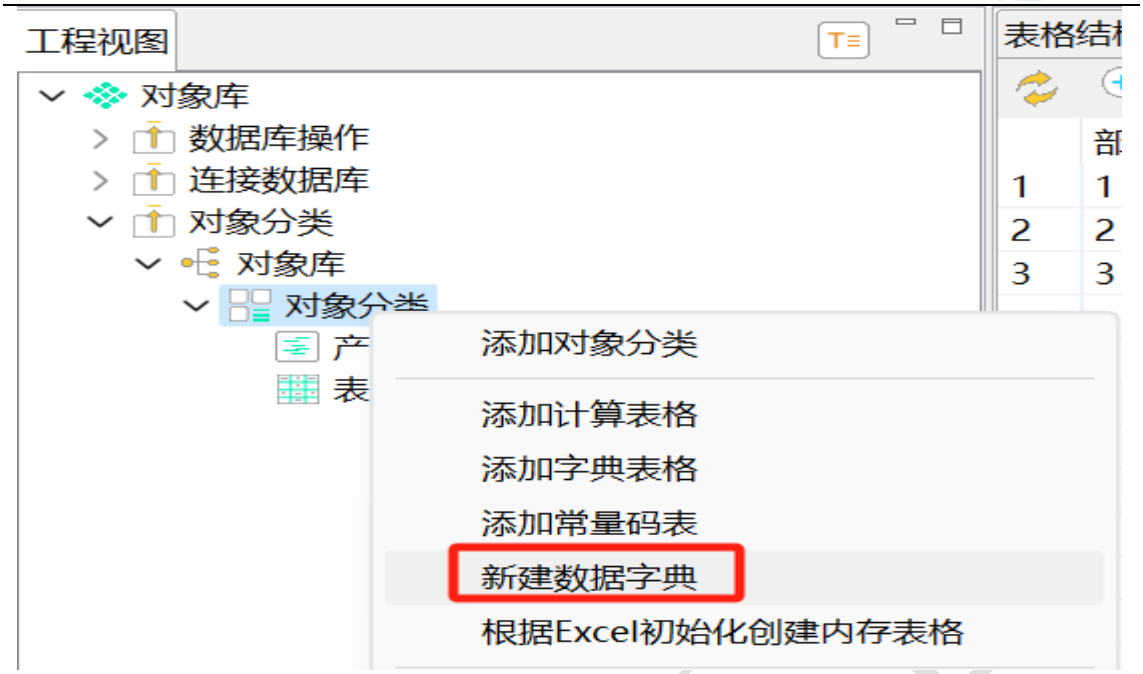


点击“表格”，切换至“表格数据”查看表数据：

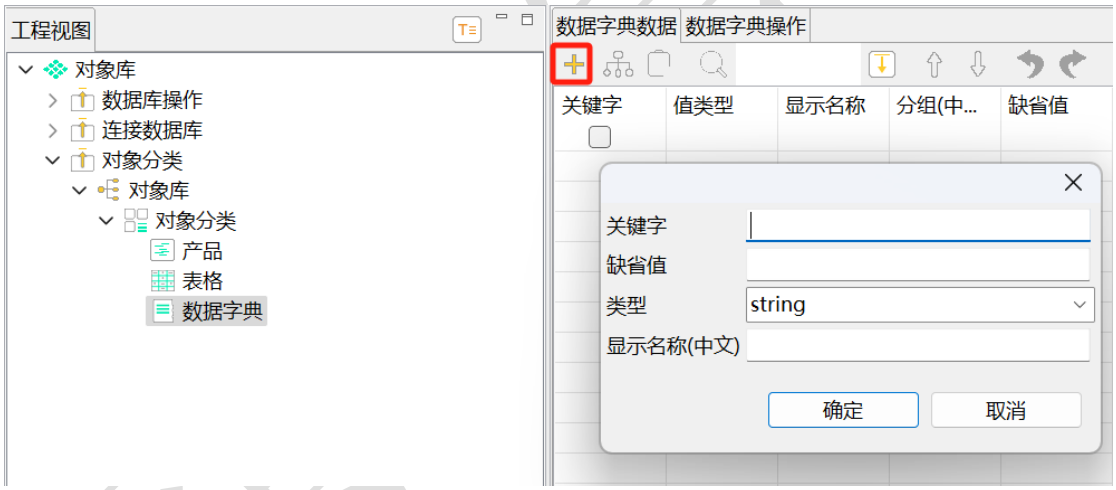


(3) 数据字典

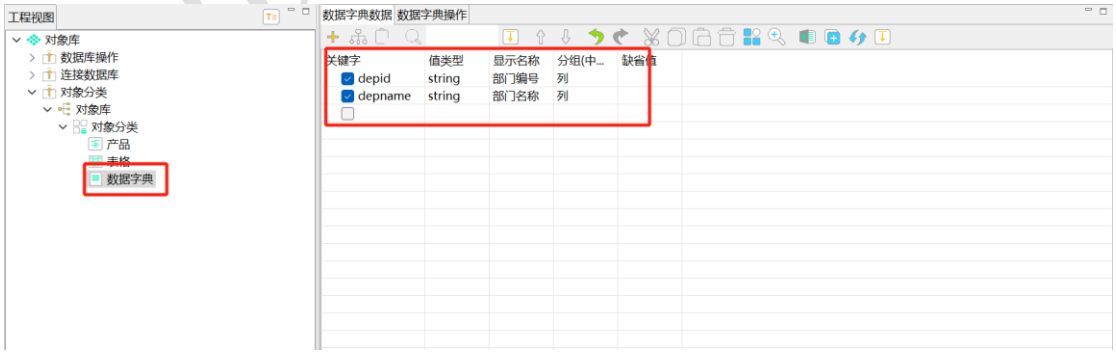
选择对象库，右键新建数据字典：



选择新建的数据字典，在中间编辑窗体中点击工具栏中的“ ”添加数据“部门编号（depid）、部门名称（depname）”：



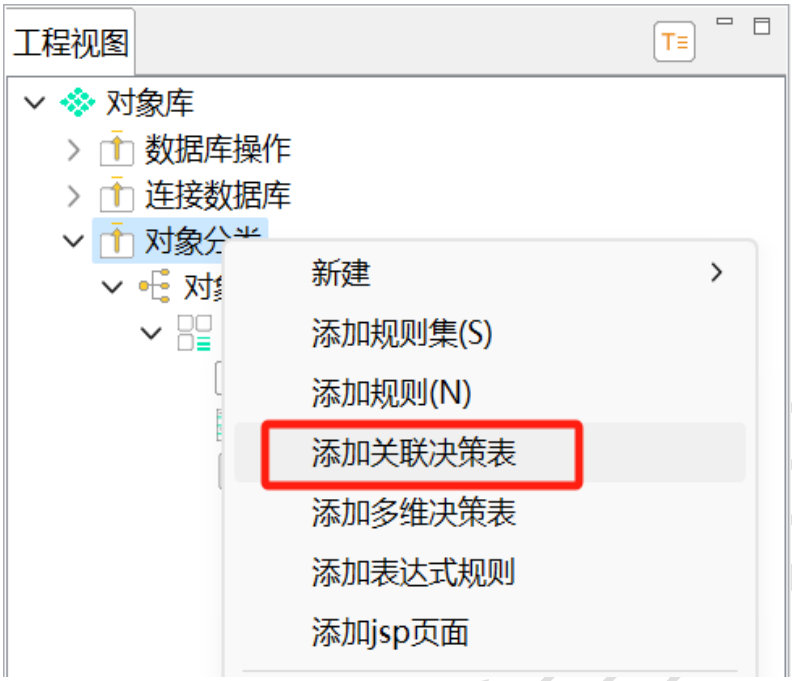
根据上述步骤添加“部门名称（depname）”，完成后数据字典数据如下图所示：



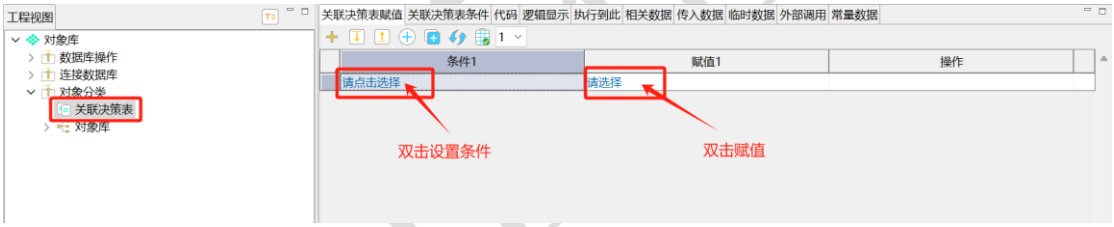
4．编写规则

（1）添加关联决策表

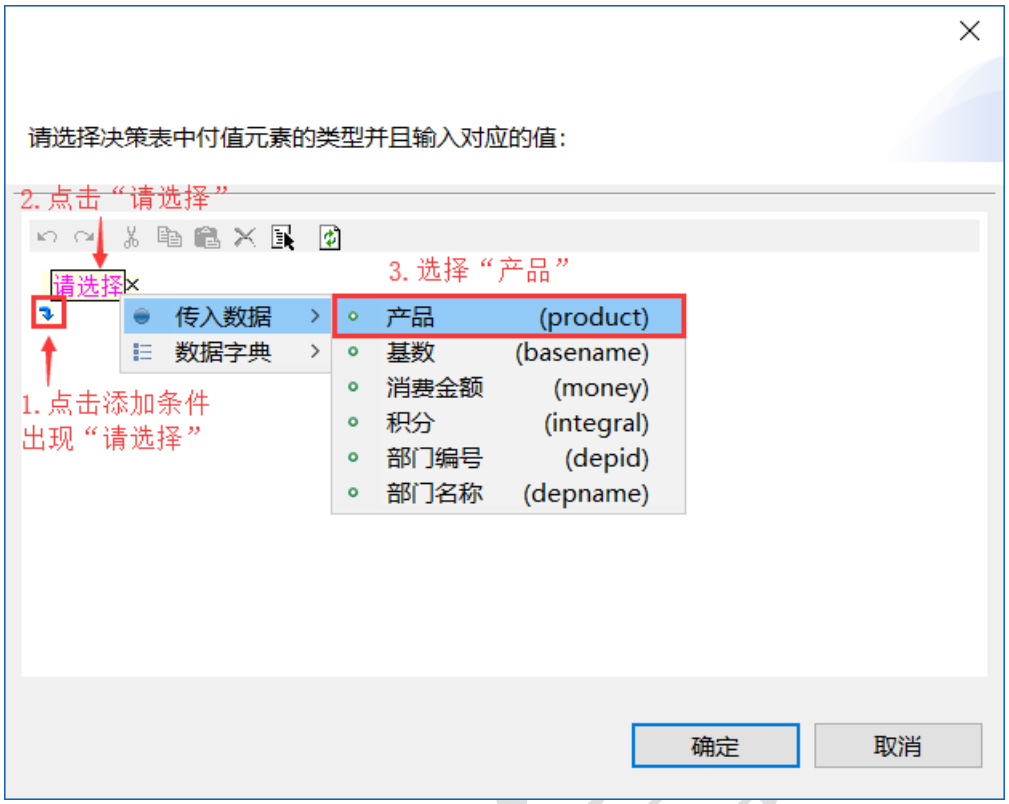
在“对象分类”规则包下添加关联决策表。右键规则包，选择添加关联决策表：



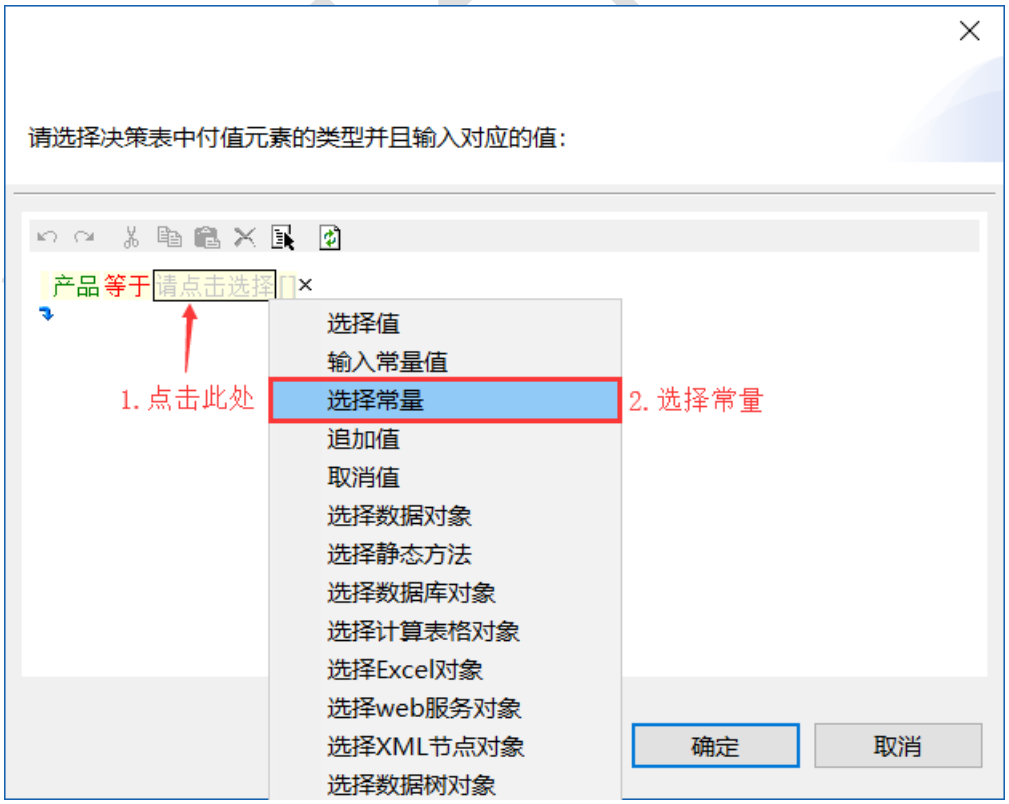
打开“关联决策表”，对其进行赋值：

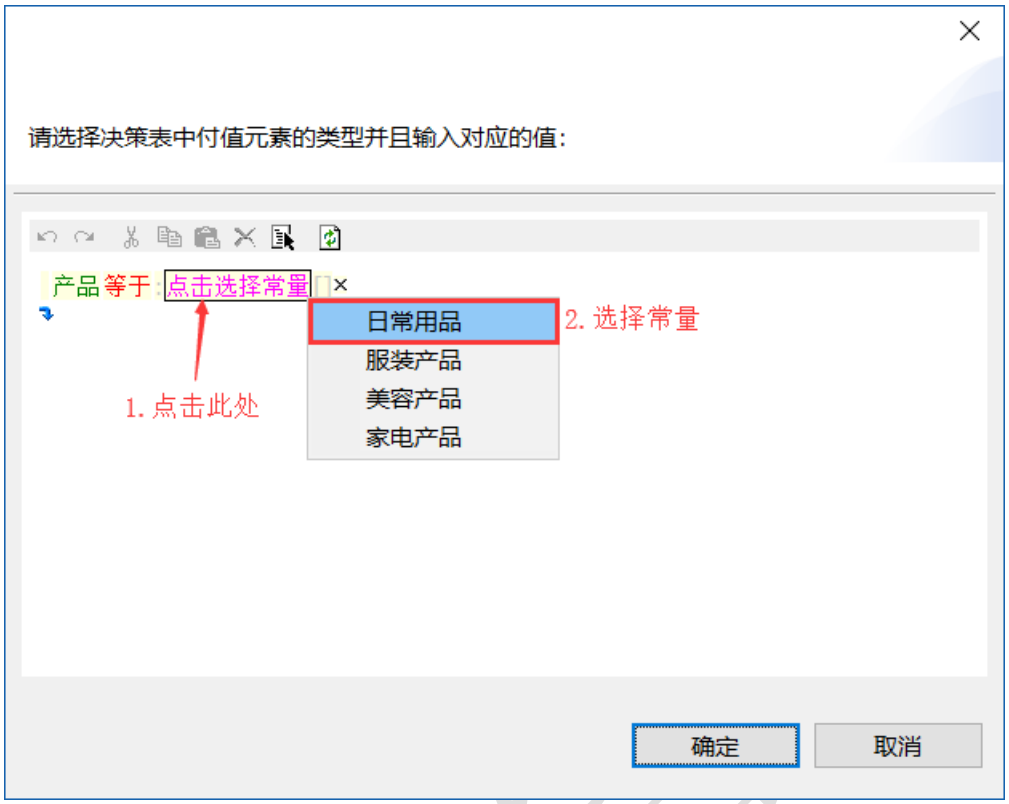


在弹出框中根据以下步骤添加条件：

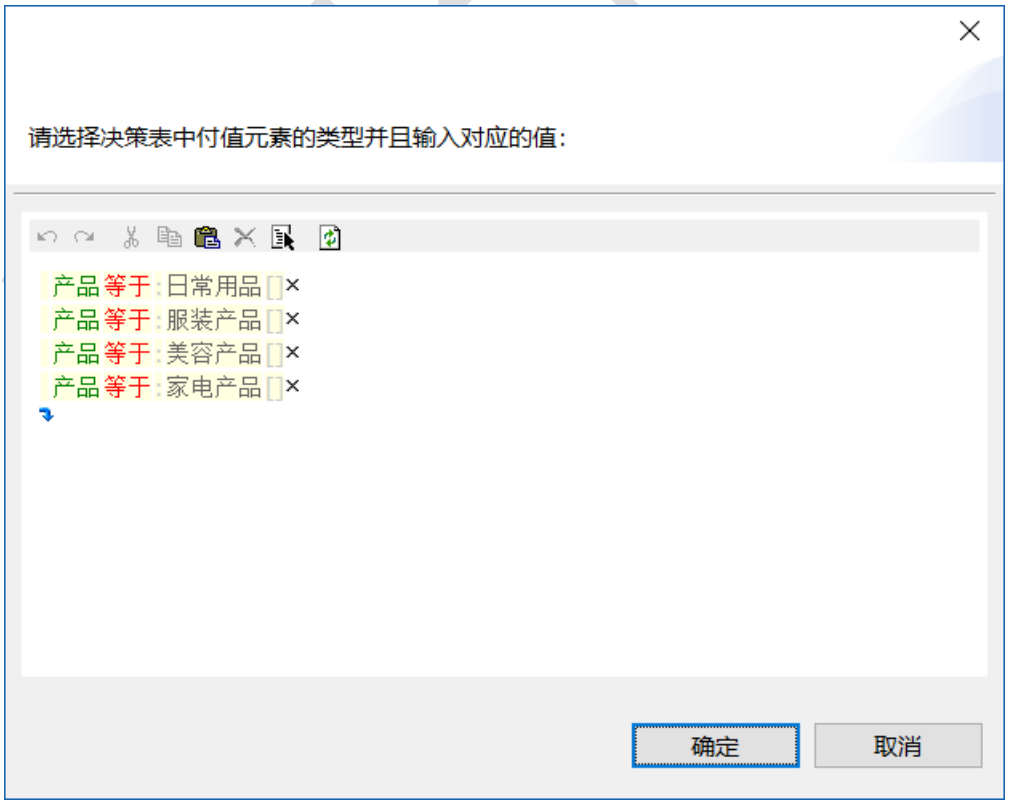


对变量“产品”赋值，如下所示：

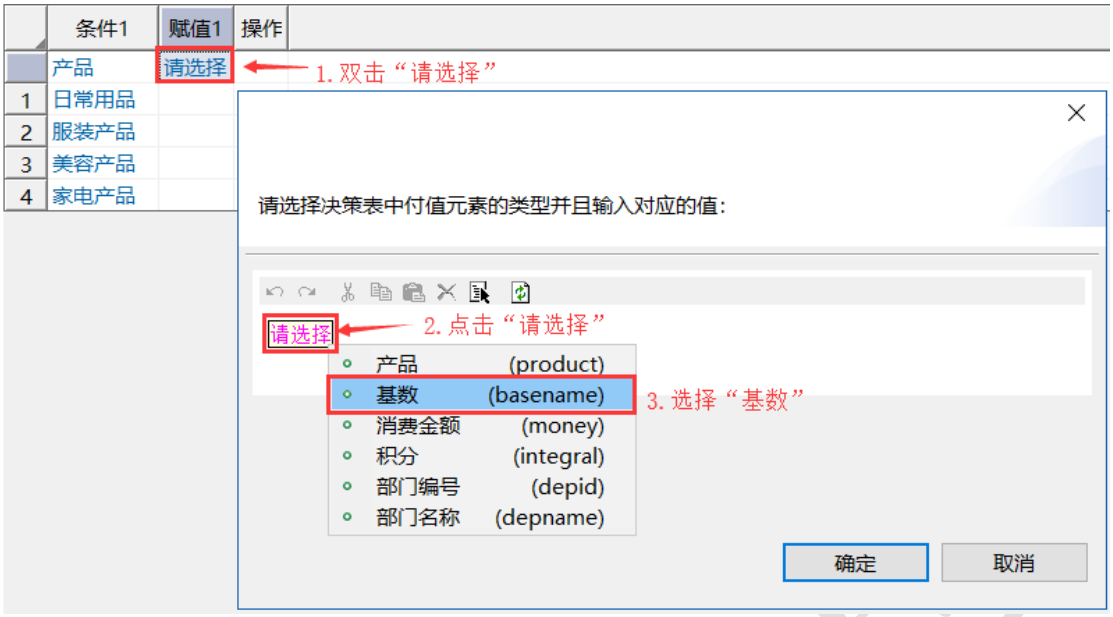




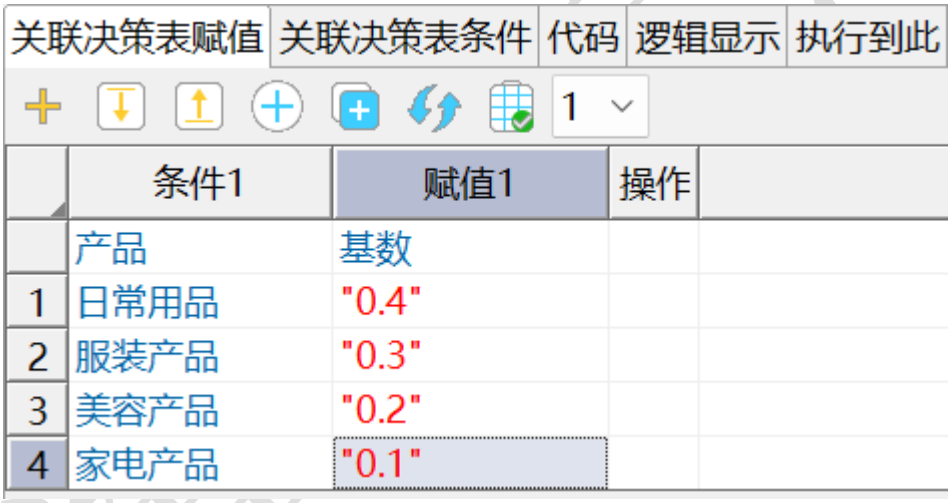
根据上述步骤继续添加多条条件，完成后如下图所示：



有了条件之后需要添加赋值，首先添加赋值元素，过程如下：



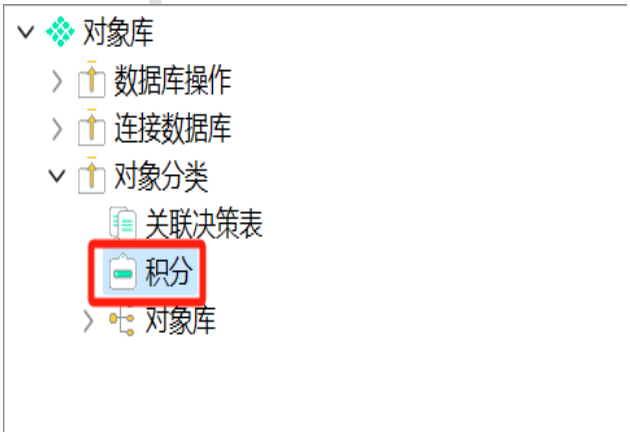
为赋值元素“基数”赋值，双击“基数”列下方的单元格，输入常量值：



(2) 添加规则

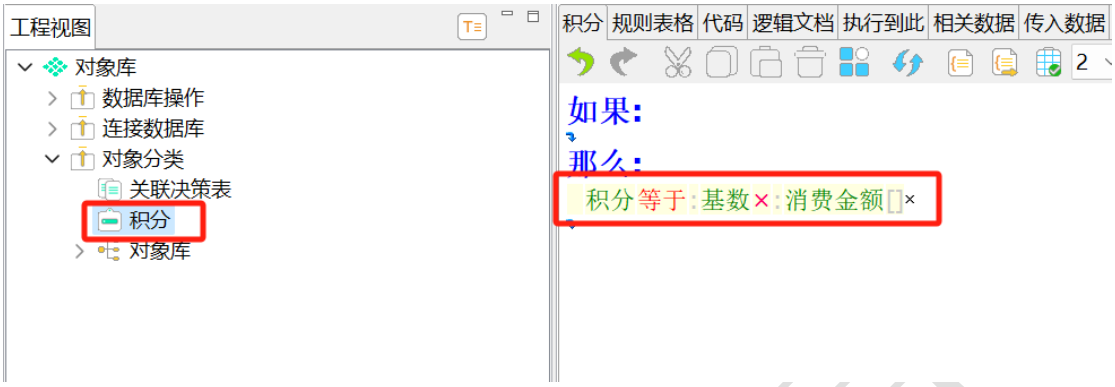
1 添加规则“积分”

在规则包下添加“积分”规则计算积分：



在中间编辑窗体中添加动作。规则为“积分=基数×消费金额”：

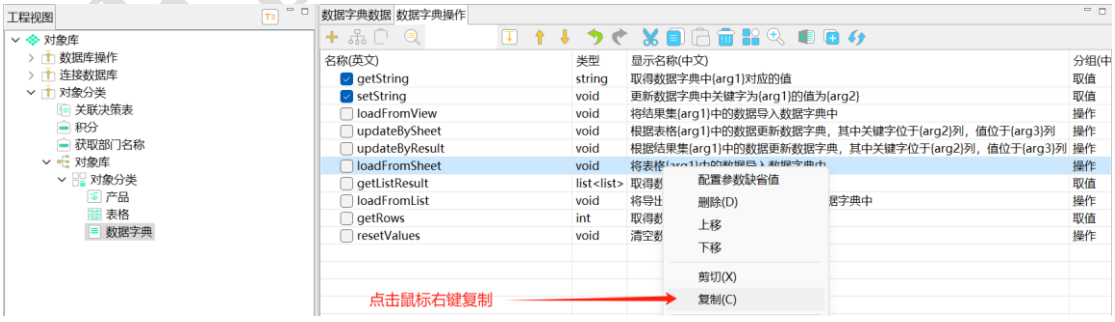
规则编写完成后如下图：



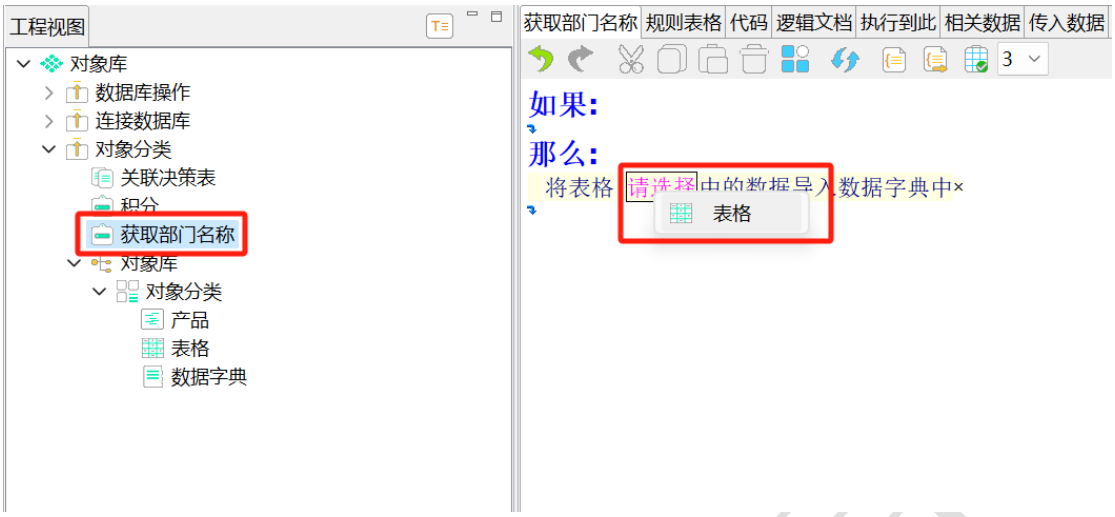
2　　添加规则“获取部门名称”

在规则包下再次添加规则，命名为“获取部门名称”：

在规则“获取部门名称”中添加动作，首先要将内存表“表格”中的数据全部导入到“数据字典”中。选择“数据字典”，在“数据字典操作”中复制“将表格{arg1}中的数据导入数据字典中”：

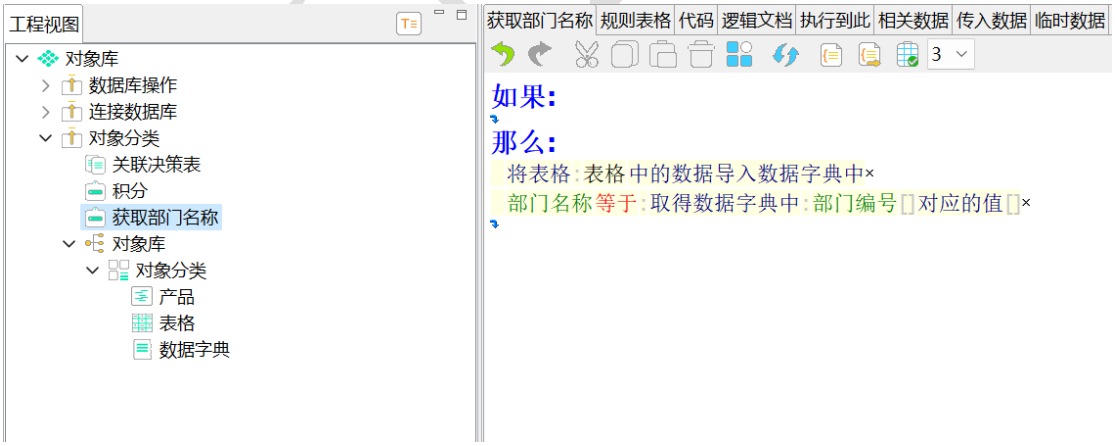


将复制的方法粘贴到规则“获取部门名称”中，并对{arg1}赋值：

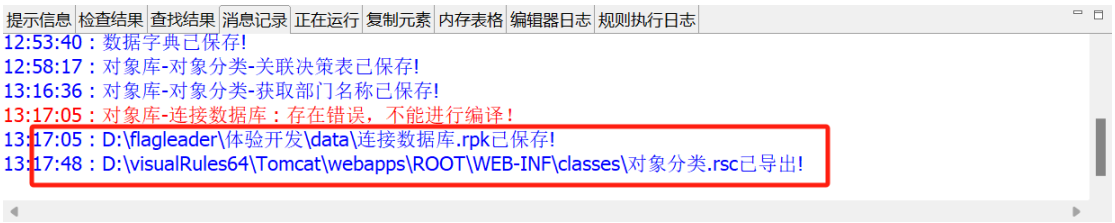


根据传入数据中的“部门编号”去取“数据字典”中对应的部门名称值：

完成后如下图：

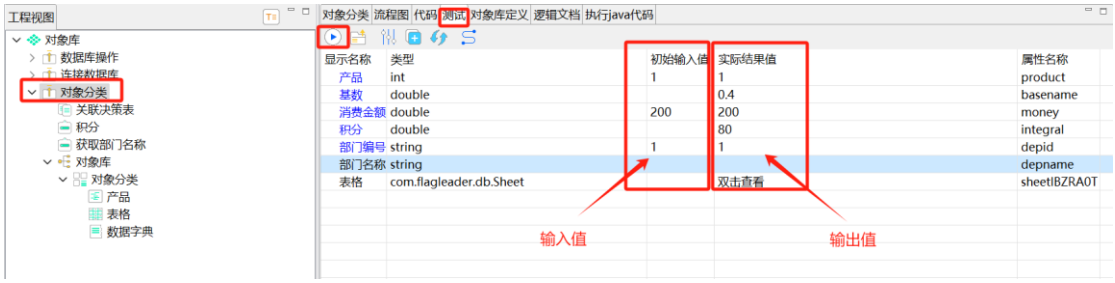


点击工具栏中的全部保存按钮“  ”保存并编译，保存之后在消息窗口中会出现如下提示（其中 rpk 是规则保存文件，可用规则配置器直接打开；rsc 是规则编译后的二进制文件，可将其放在程序中进行调用）：



5．测试

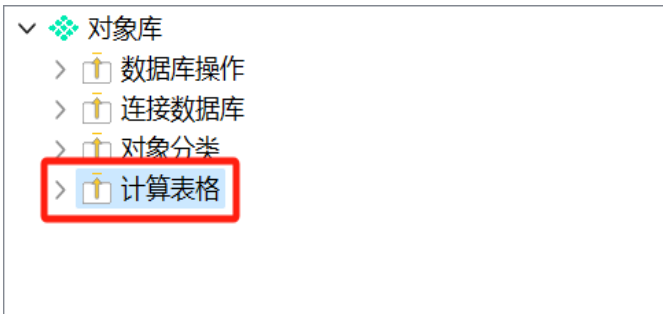
选择规则包“对象分类”，在中间编辑窗口中切换至测试，输入如下初始值进行测试：



2.3.3. 计算表格

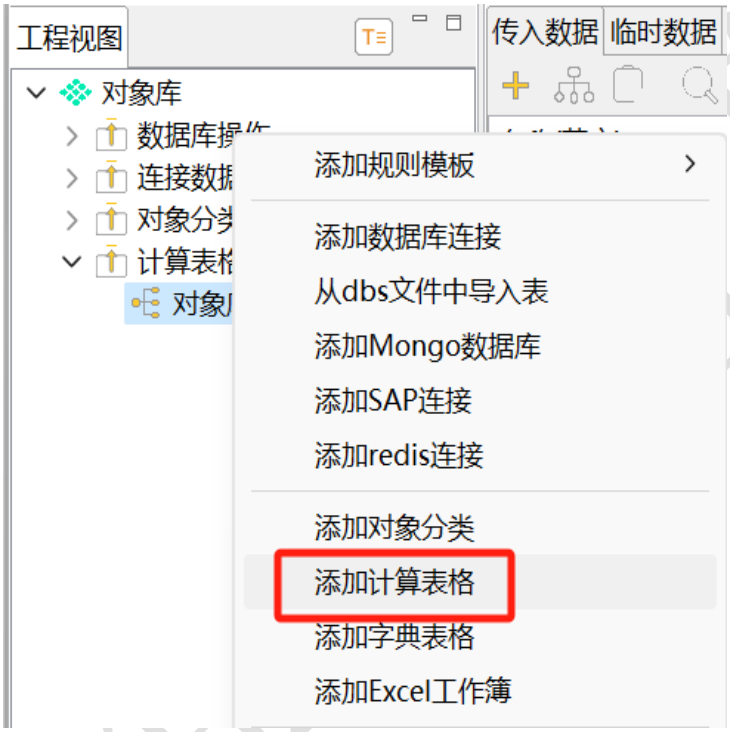
1．创建规则包

在“对象库”工程下新建名为“计算表格”的规则包：

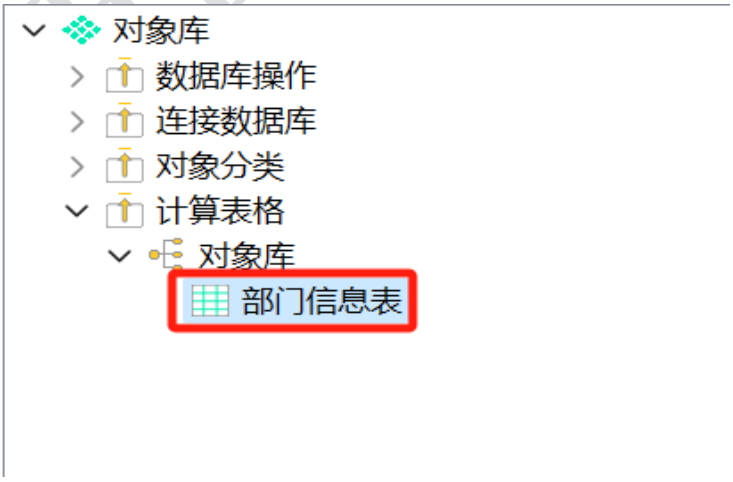


2．添加计算表格

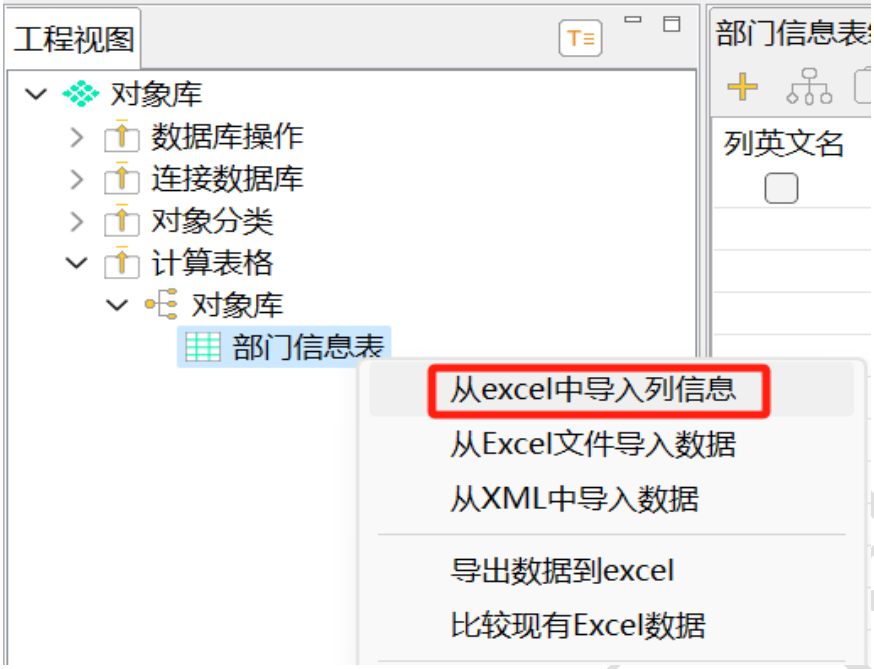
打开“计算表格”规则包，在“对象库”下添加计算表格：



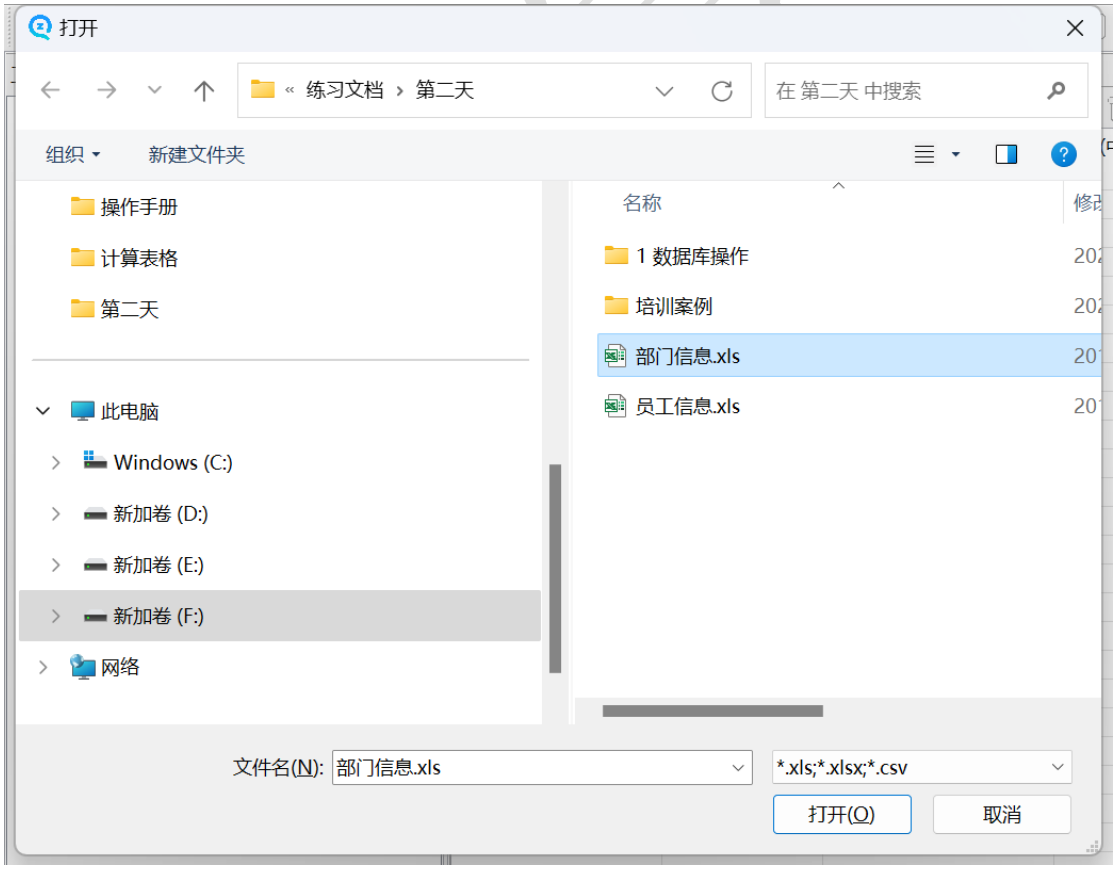
将其命名为“部门信息表”。



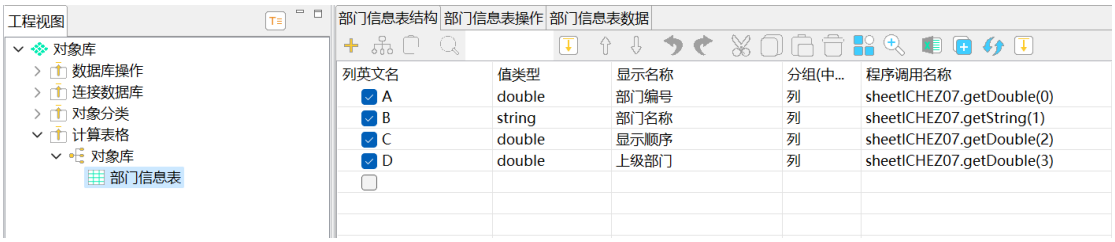
右键“部门信息表”选择“从 excel 中导入列信息”：



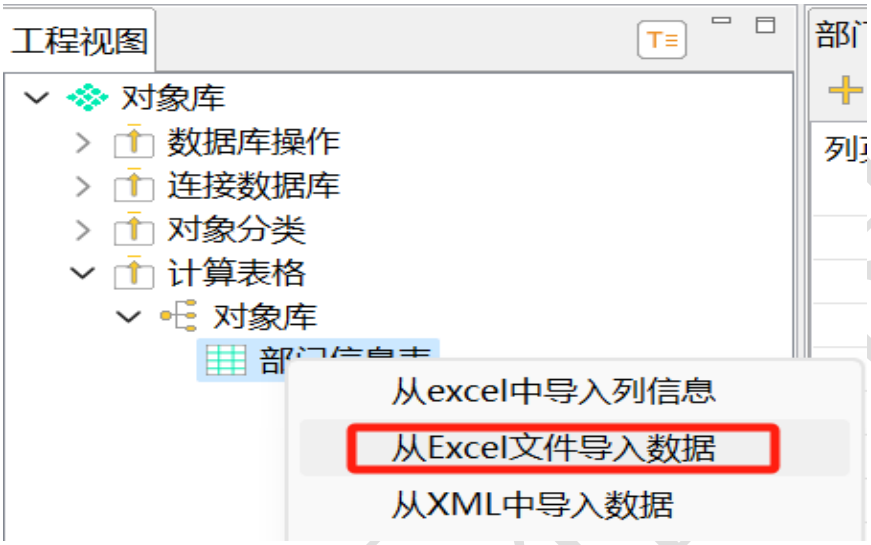
弹出对话框如图：



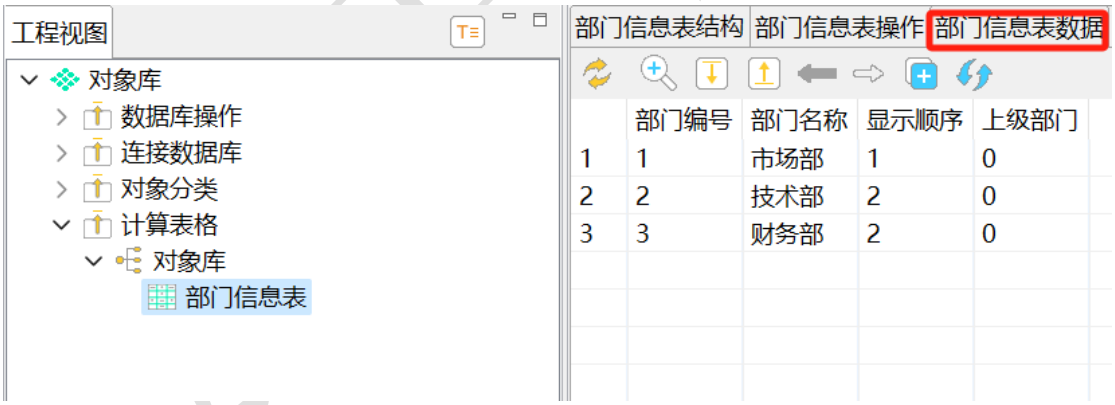
点击“部门信息表”查看“部门信息表结构”：



有了列字段之后，需要给表格添加数据，右键“部门信息表”选择“从 Excel 文件导入数据”：



完成之后，点击“部门信息表”查看“部门信息表数据”：

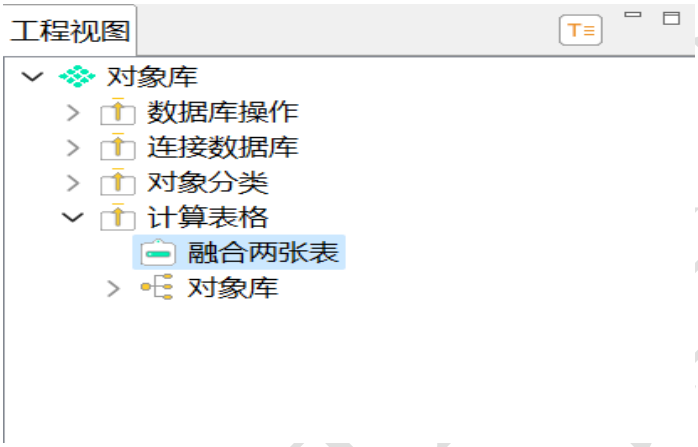


同上，再次添加一个名为“员工信息表”的计算表格，并导入列和数据：

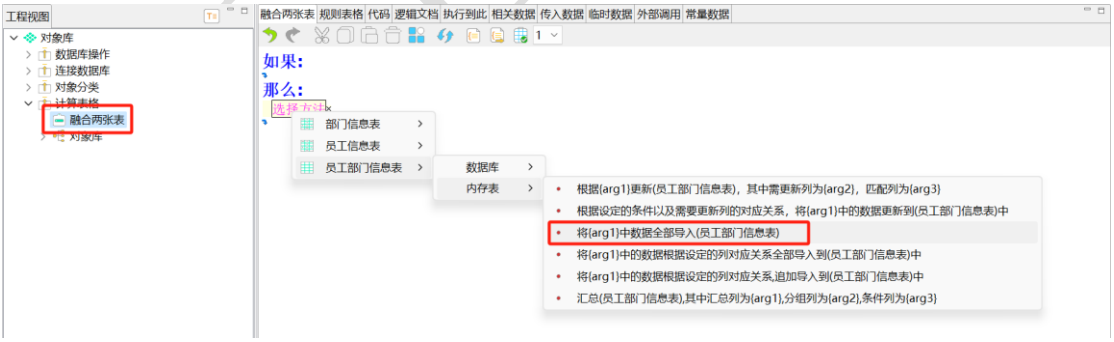
最后再添加一个名为“员工部门信息表”计算表格，并添加如下表字段：

3．编写规则

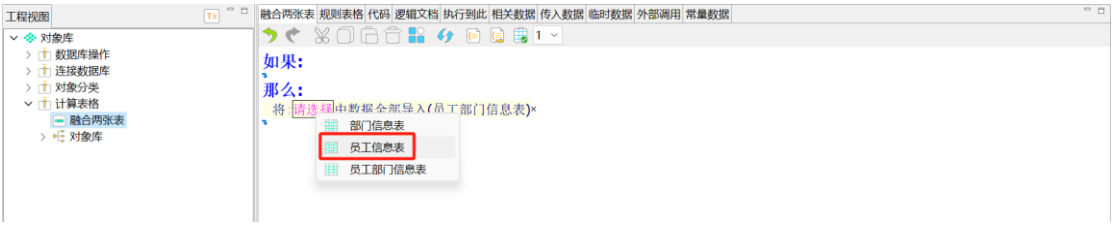
点击“计算表格”规则包，添加名为“融合两张表”的 规则：



点击添加动作，选择“执行方法”，选择“员工部门信息表”的“内存表”下的“将{arg}中的数据全部导入（员工部门信息表）”方法：



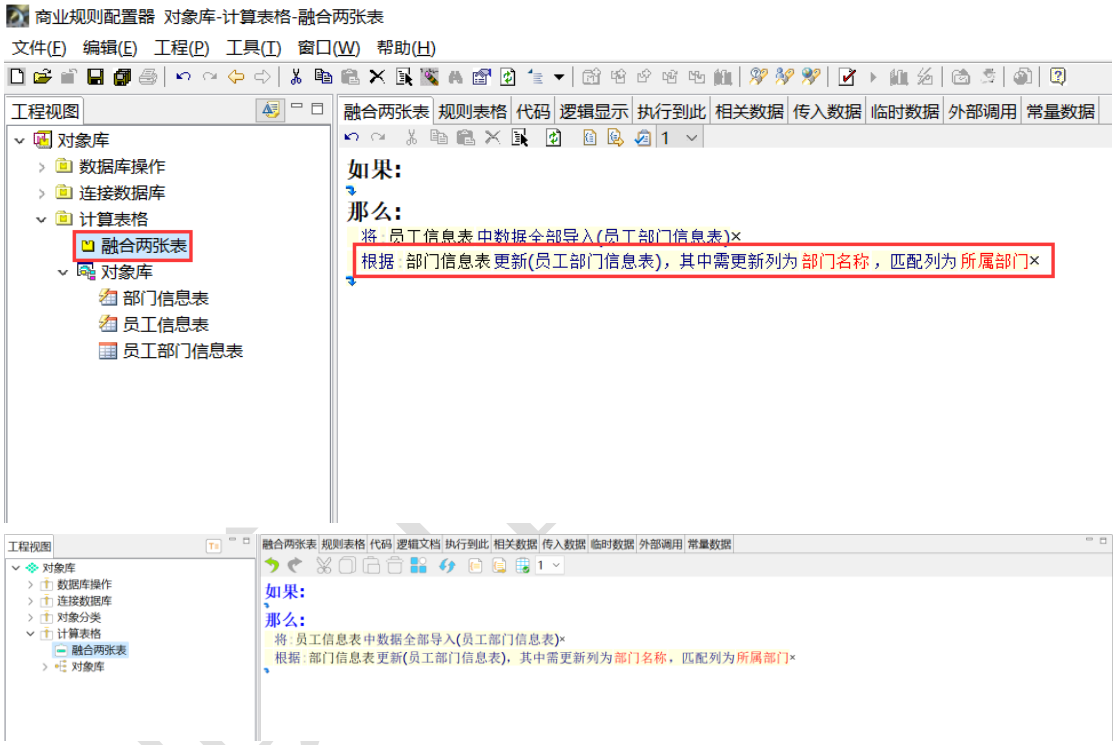
导入员工信息表中的数据，点击选择“员工信息表”：



有了“员工信息表”中的数据，再根据“depid”融合部门名称：

点击添加动作，选择“执行方法”，选择“员工部门信息表”的“内存表”下“根据{arg1}更新（员工部门信息表），其中需更新列为{arg2}，匹配列为{arg3}”：

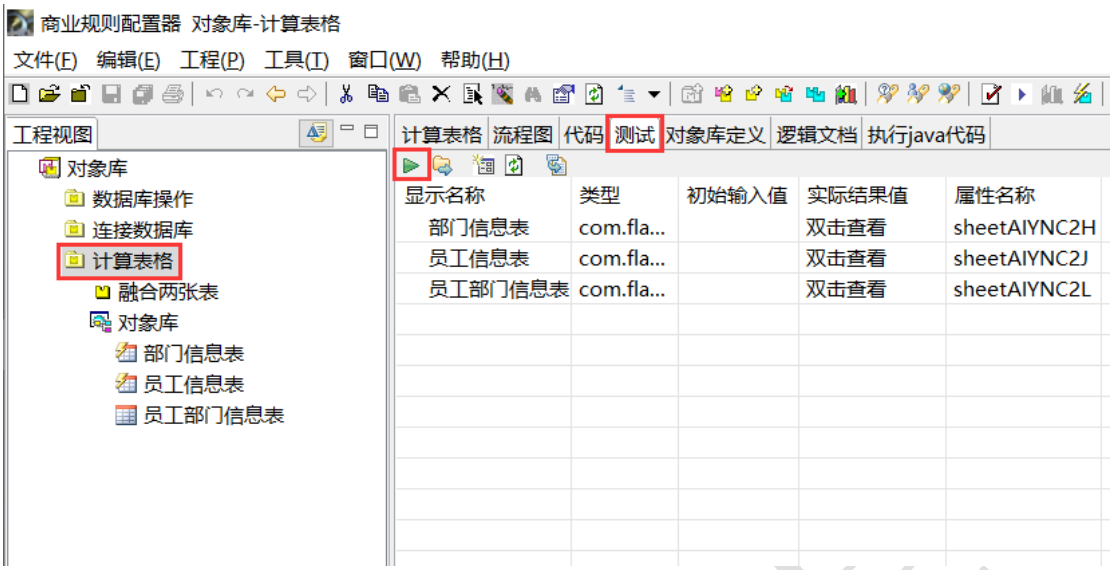
选择根据“部门信息表”更新，更新列为“部门名称”，匹配列为“所属部门”：



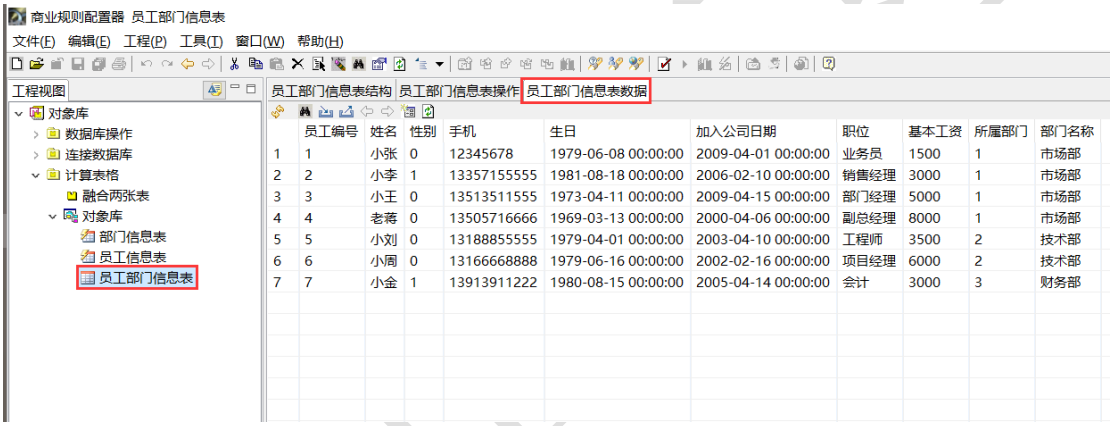
完成之后保存并编译。

4．测试

点击“计算表格”规则包，在测试中点击执行：



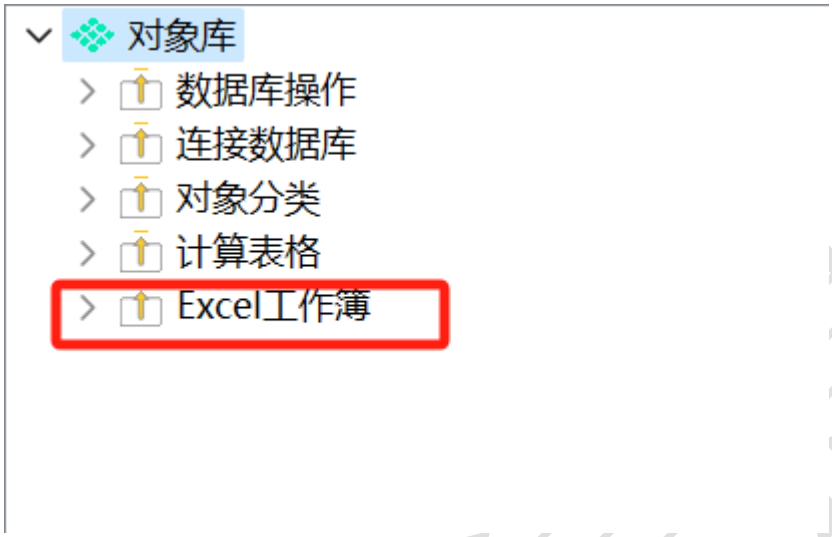
点击“员工部门信息表”查看“员工部门信息表数据”：



2.3.4. Excel 工作簿

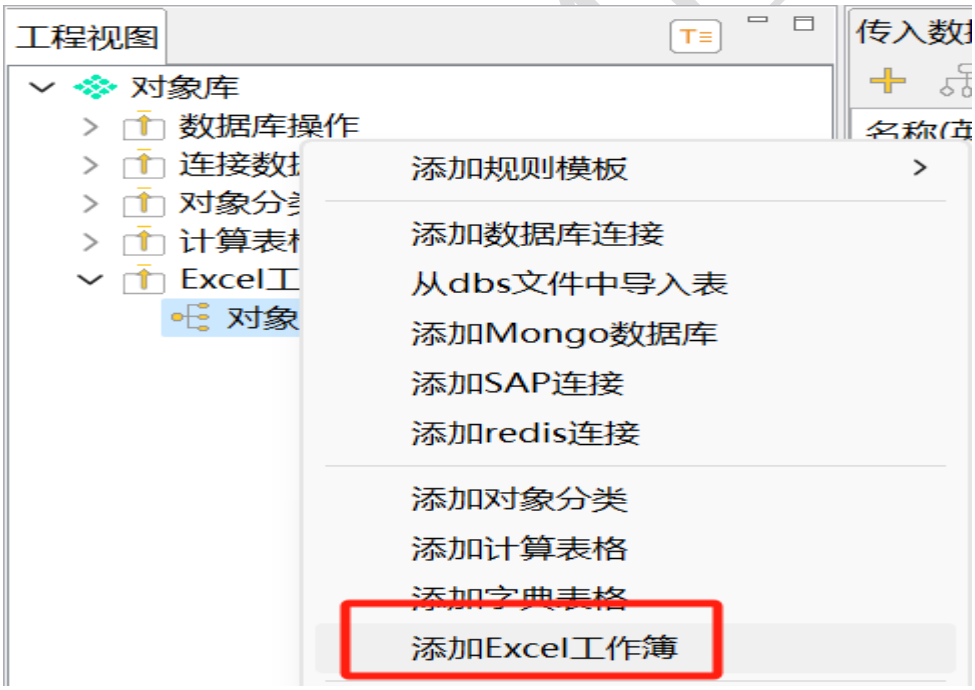
1. 创建规则包

在“对象库”工程下新建名为“Excel 工作簿”的规则包：

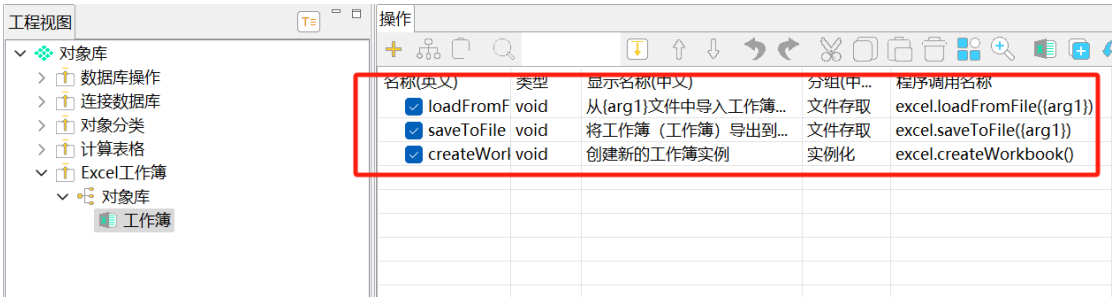


2. 添加 Excel 工作簿

打开“Excel 工作簿”规则包，在对象库下右键“添加 Excel 工作簿”：

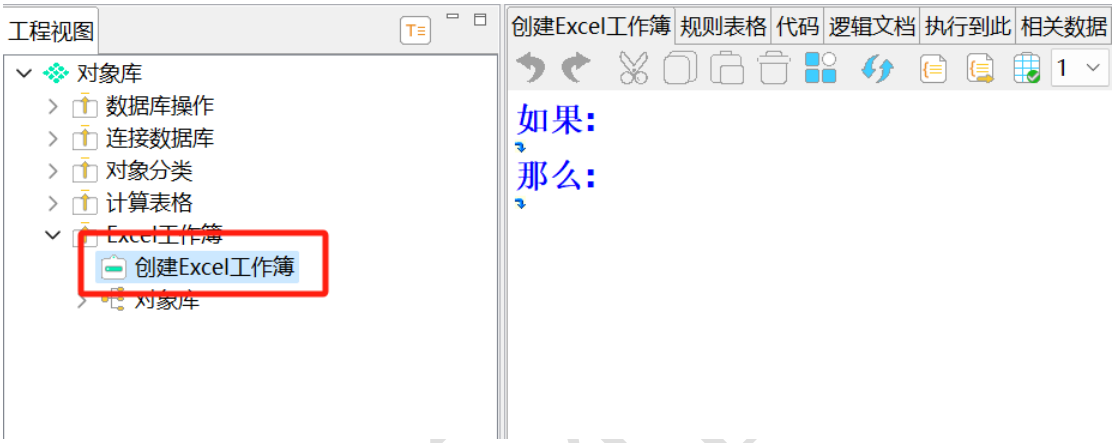


点击确定，生成 Excel 工作簿，打开 Excel 工作簿在“操作”下会看到如下方法：

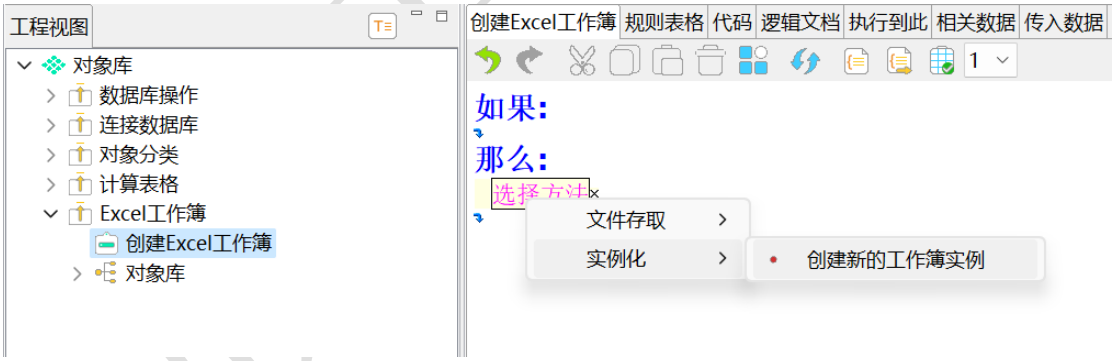


3．新建规则

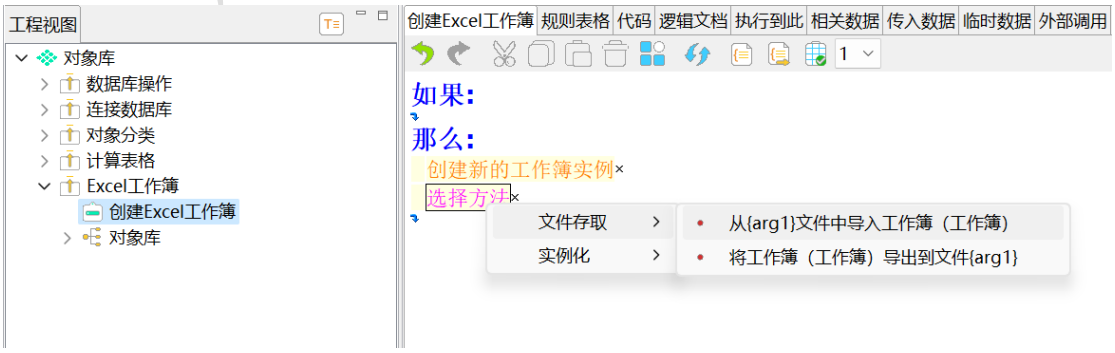
点击“Excel 工作簿”规则包，右键添加名为“创建 Excel 工作簿”的规则：



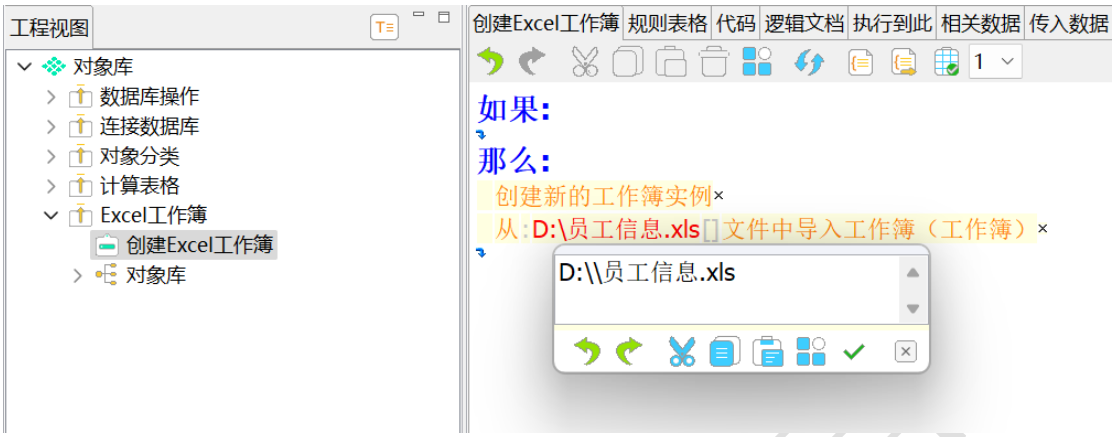
在右侧点击“那么”下的“”添加动作，“点击请选择”→“执行方法”→“实例化”→“创建新的工作簿实例”



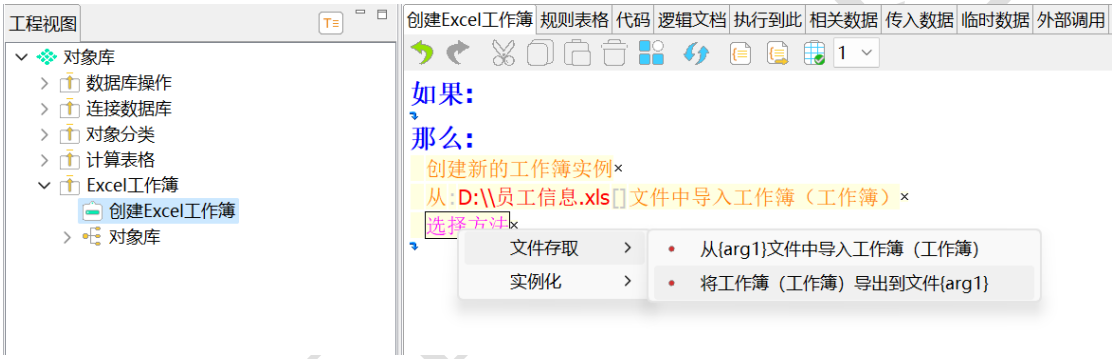
再次点击“”添加动作，“点击请选择”→“执行方法”→“文件取存”→“从{arg1}文件中导入工作簿（工作簿）”



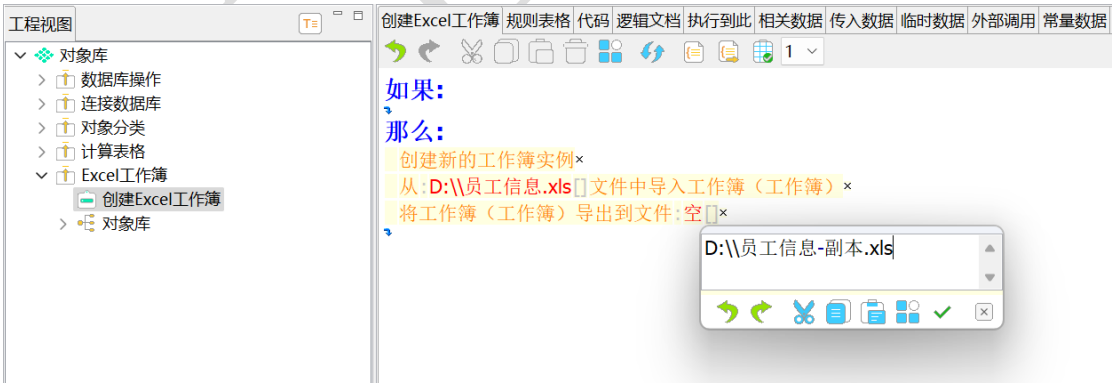
点击“点击请选择[]”选择“输入常量值”，点击“空”，在弹出框中输入 Excel 文件所在路径：



根据上面的方法，将“文件取存”中的“将工作簿（工作簿）导出到文件{arg1}”添加到规则中：



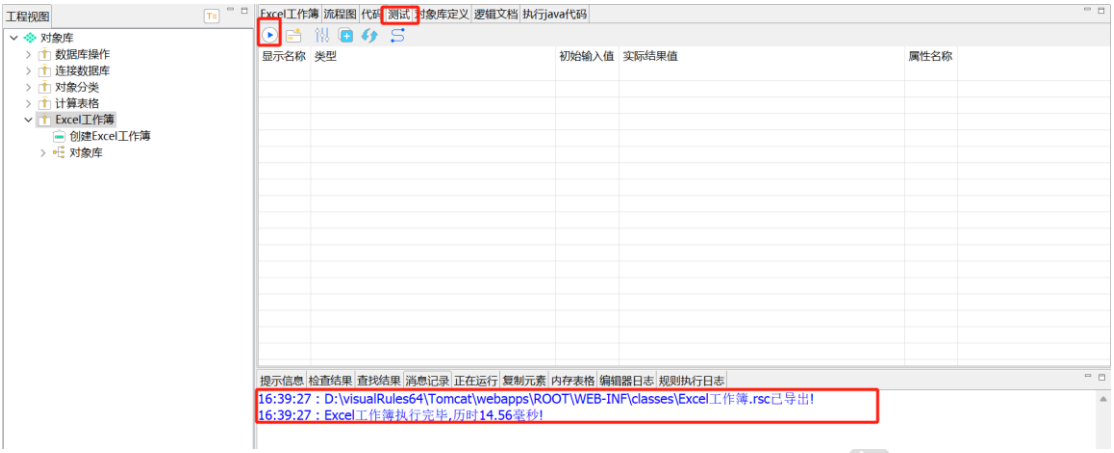
点击“点击请选择[]”选择“输入常量值”，点击“空”输入文件输出路径：



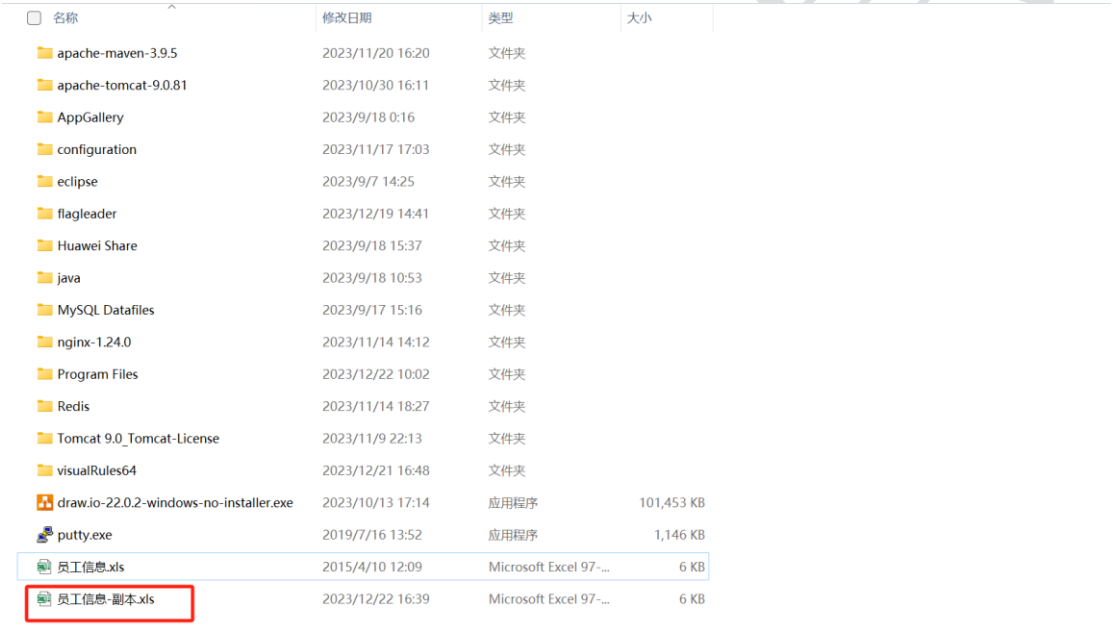
完成后保存并编译。

4．测试

选择“Excel 工作簿”点击右侧“测试”，在测试界面中点击“▶”执行，执行完成在消息窗口中会显示执行所用时间：



打开 D 盘，查看生成文件：

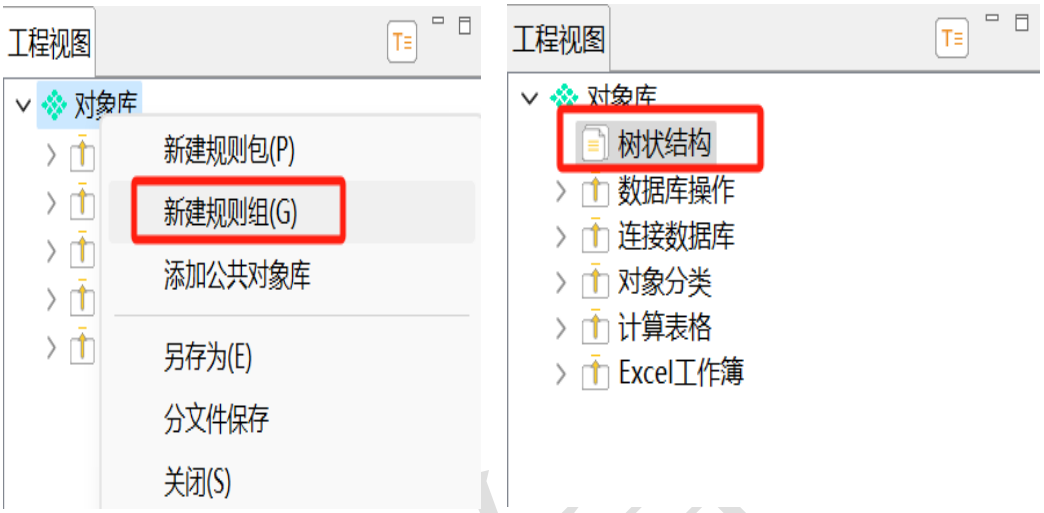


2.3.5. 树状结构

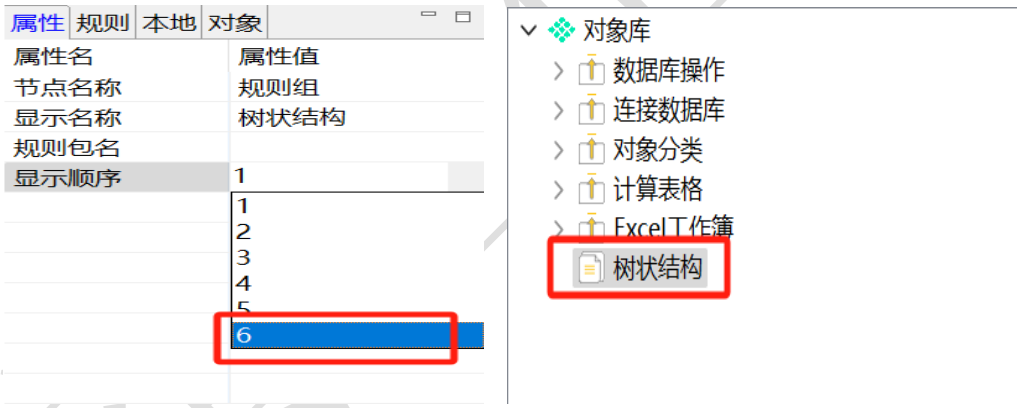
规则引擎在实际运用中往往是通过调用程序端的接口来获取数据，这个接口通常返回的是 json 或者 xml 格式的文件，而解析这种文件则要用到树状结构。

首先，创建一个规则组，然后在规则组中分别建立 xml、json 的规则包，来分别介绍其解析方法。

在工程“对象库”下创建规则组。右键“对象库”选择“新建规则组”，并重命名为“树状结构”：



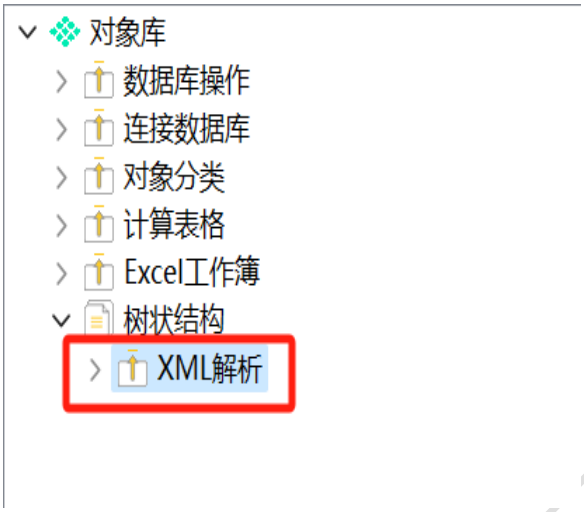
在属性窗口中更改显示顺序，使之调整到最后：



2.3.5.1. XML 解析

1．创建规则包

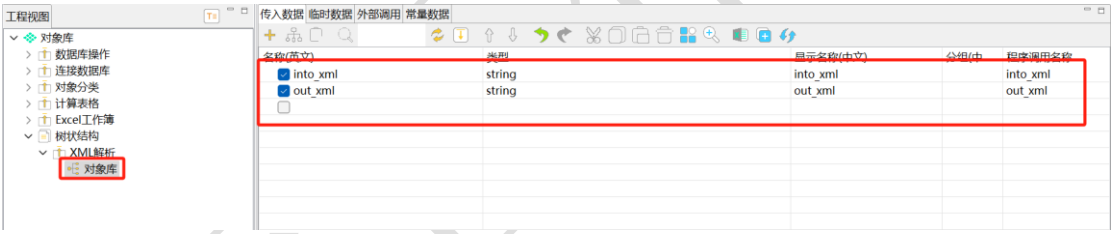
在规则组“树状结构”中，新建规则包，并将其命名为“XML 解析”：



2．创建对象库变量

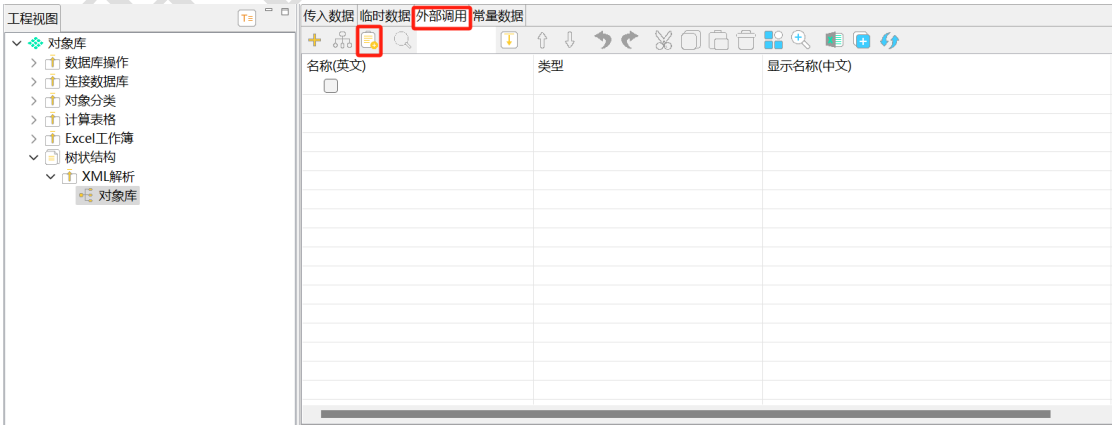
(1) 添加传入数据

打开“XML 解析”规则包，在对象库传入数据中添加如下变量：

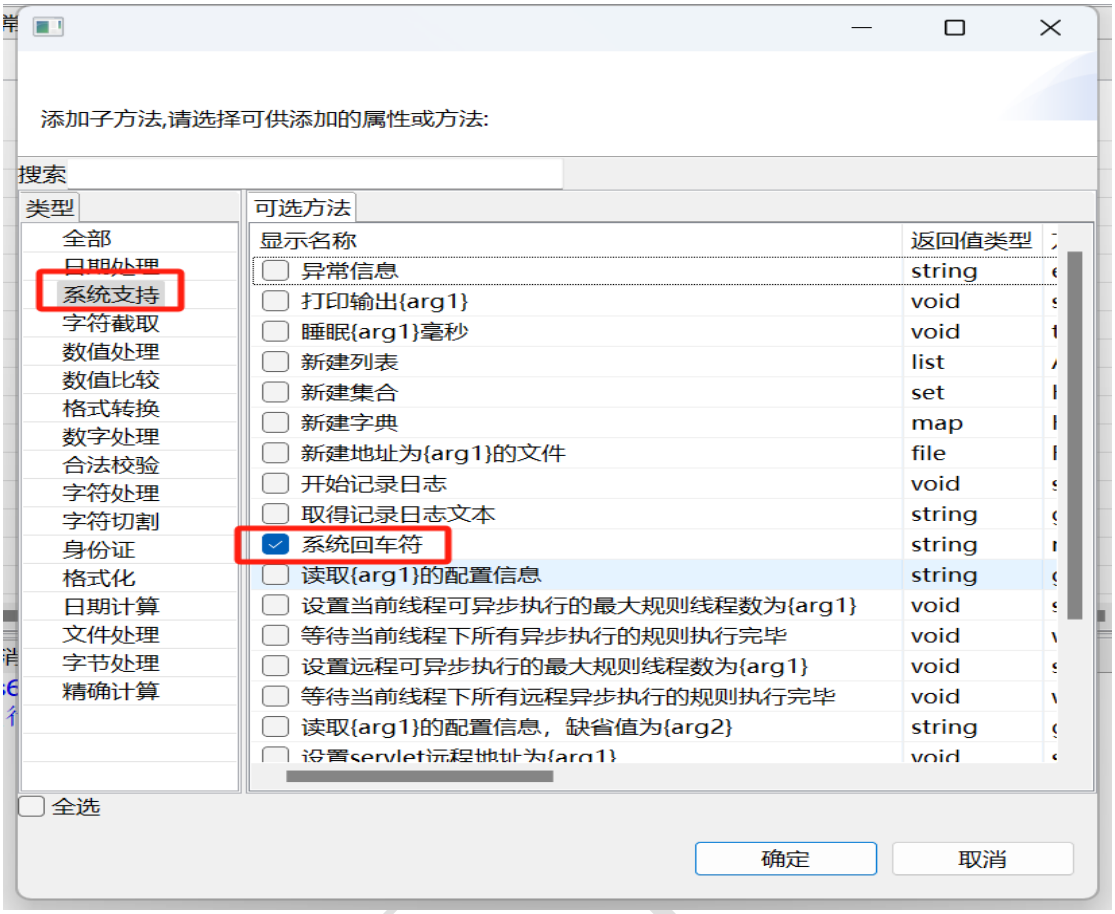


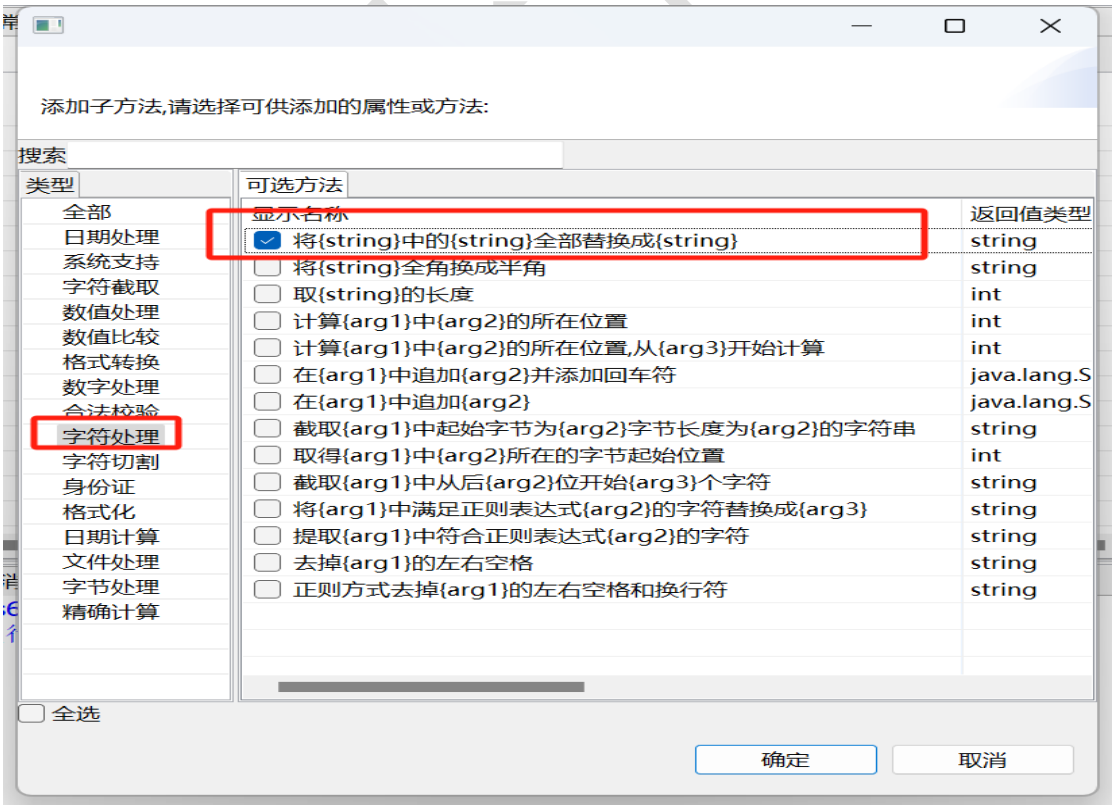
(2) 添加外部调用

在外部调用中点击“ ”添加公式：



分别添加“系统支持”中的“系统回车符”方法，和“字符处理”中的“将{arg1}中的{arg2}全部替换为{arg3}”

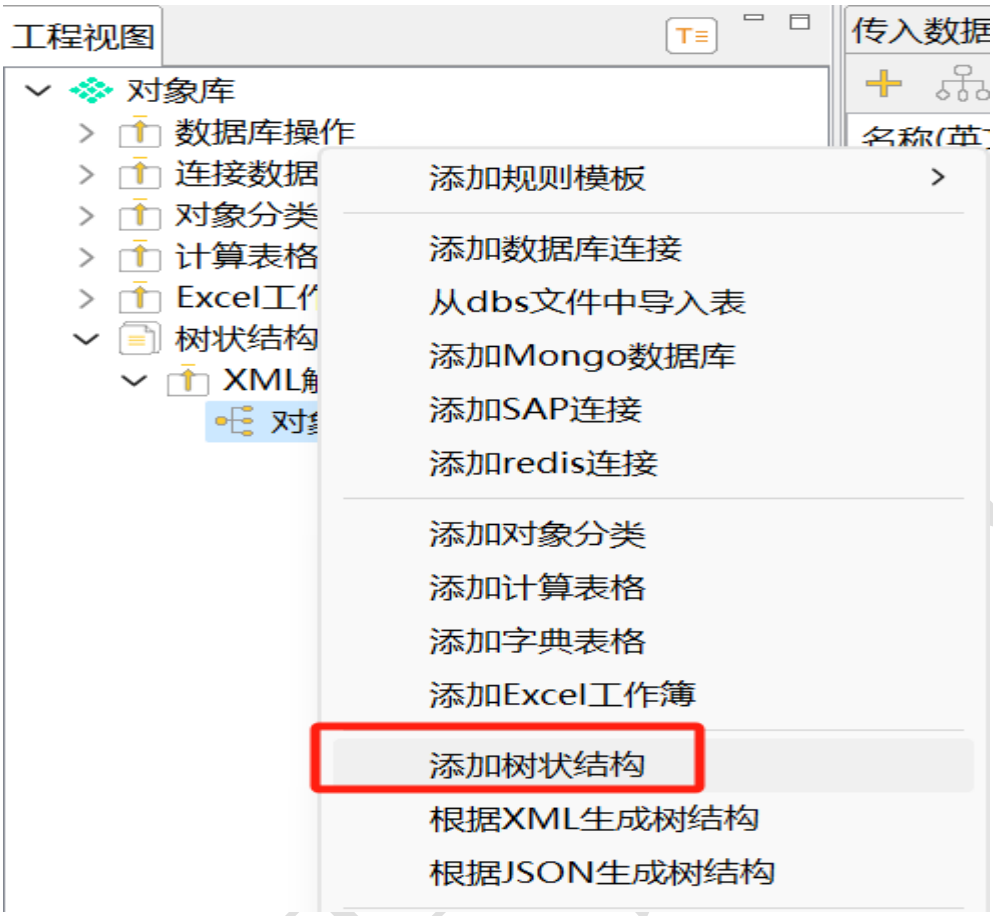




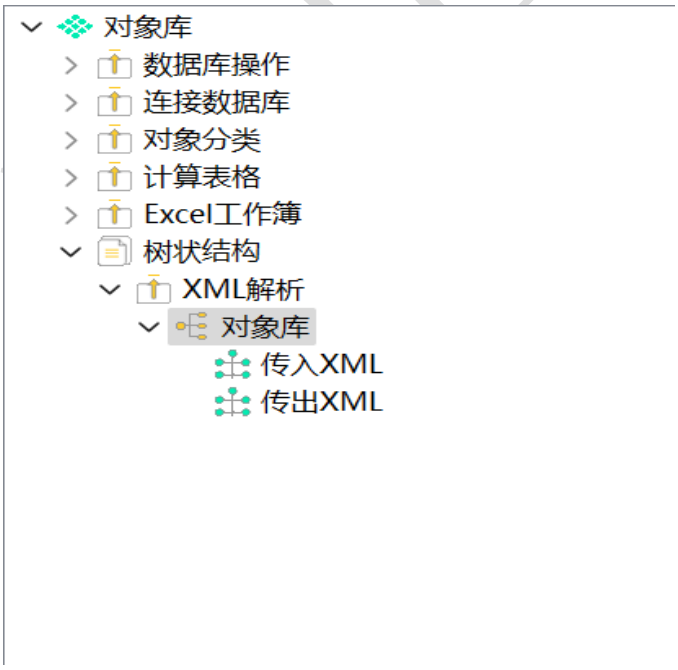
3．添加树状结构

右键对象库，选择“添加树状结构”（如果有 xml 文件，这里也可以选择“根据 XML 生

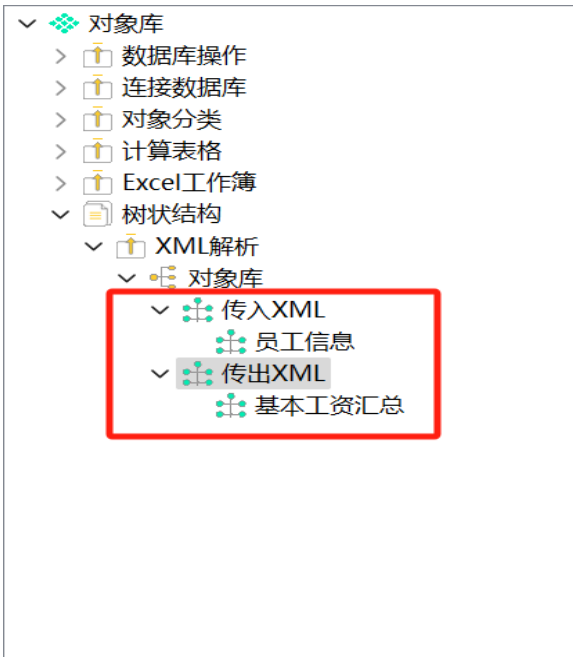
成树结构”）：



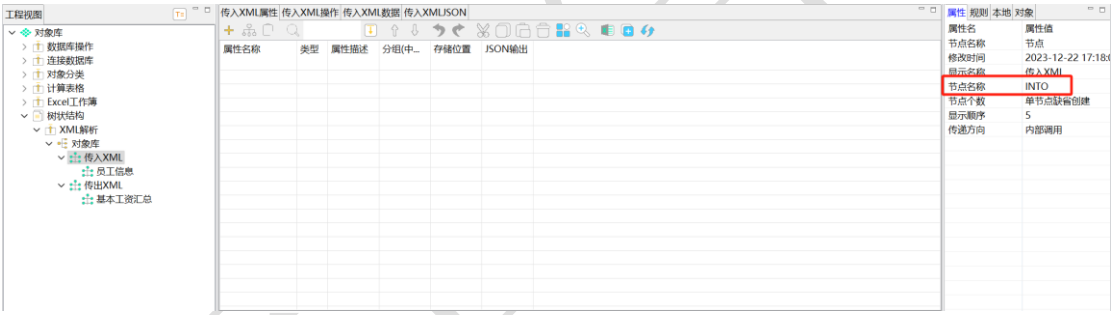
分别添加“传入 xml”和“传出 xml”为命名的树状结构：



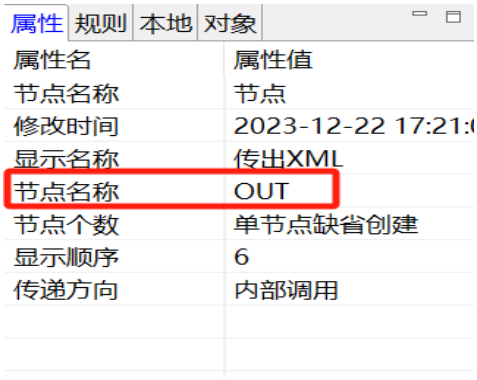
在“传入 xml”中添加“员工信息”树状结构，在“传出 xml”中添加“基本工资汇总”树状结构：



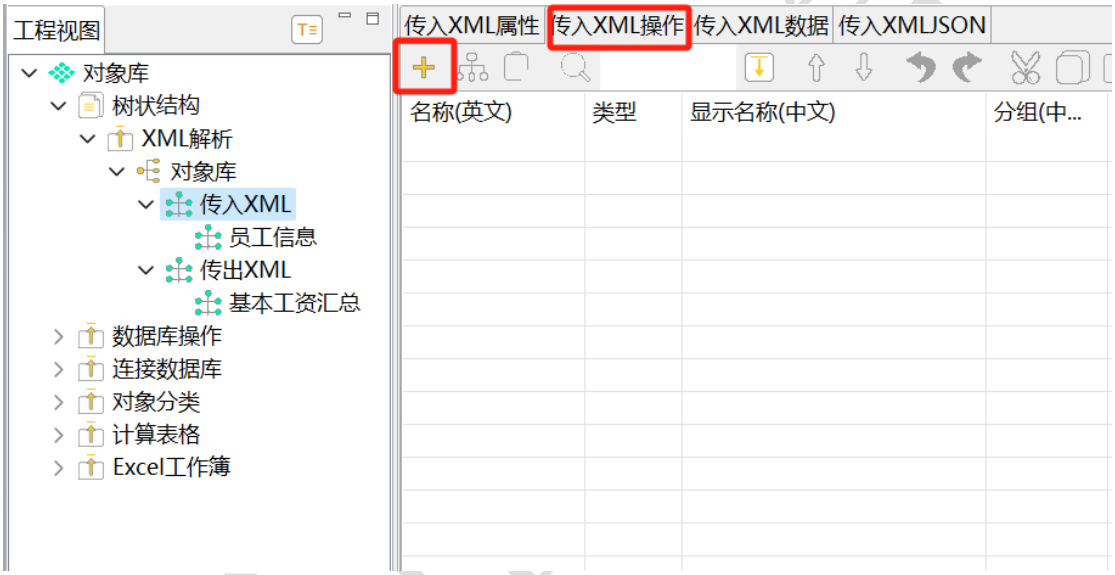
添加完成之后需要将树状结构的外部调用名统一改为英文。点击“传入 xml”，在属性编辑窗口将“节点名称”所对应的属性值改为英文，如图：



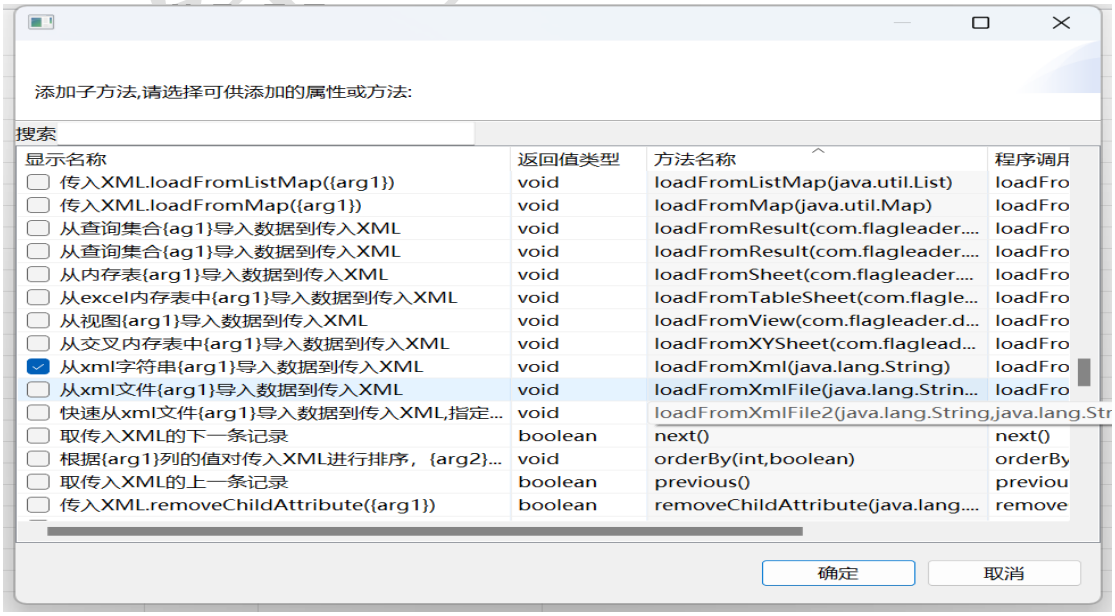
根据上述方法，将“员工信息”、“传出 xml”、“基本工资汇总”的节点名称也分别改为英文：



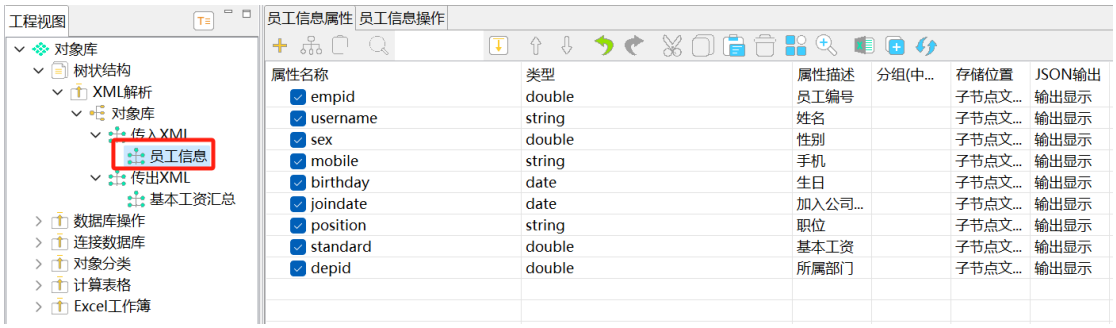
在“传入 xml”的操作中添加方法：



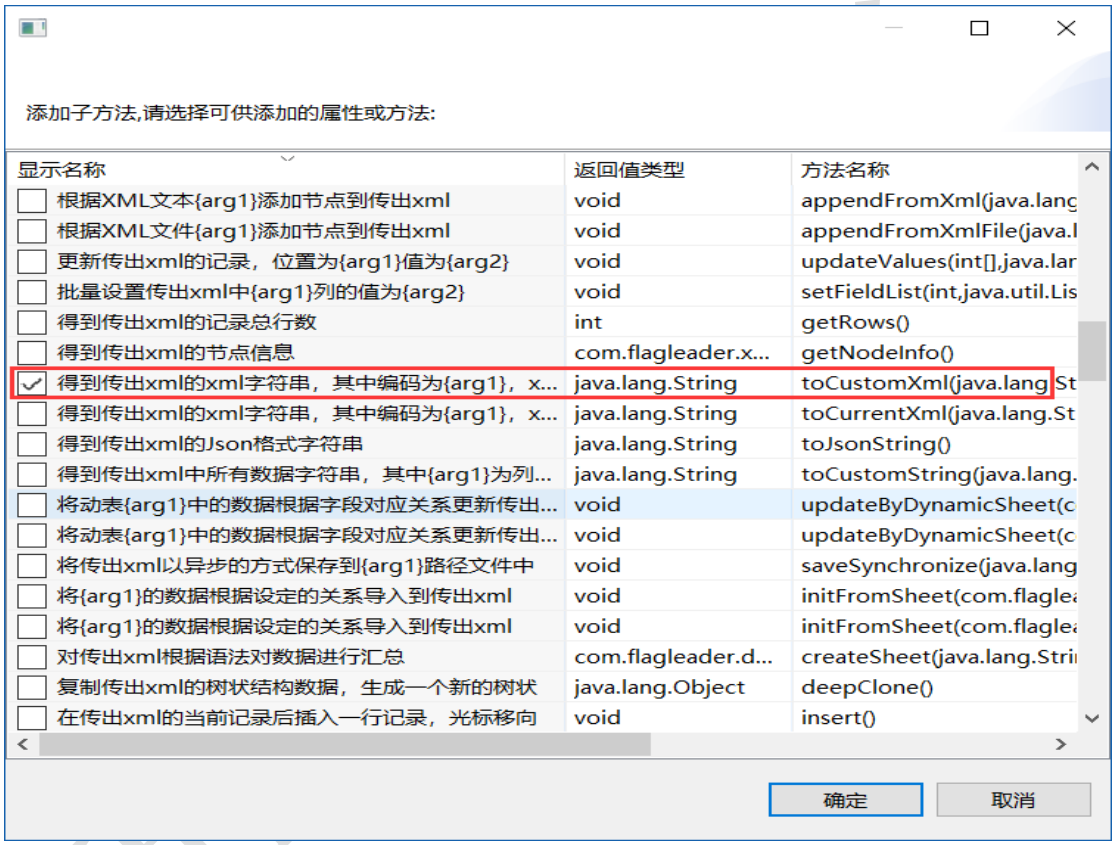
勾选此方法，点击确定添加到操作：



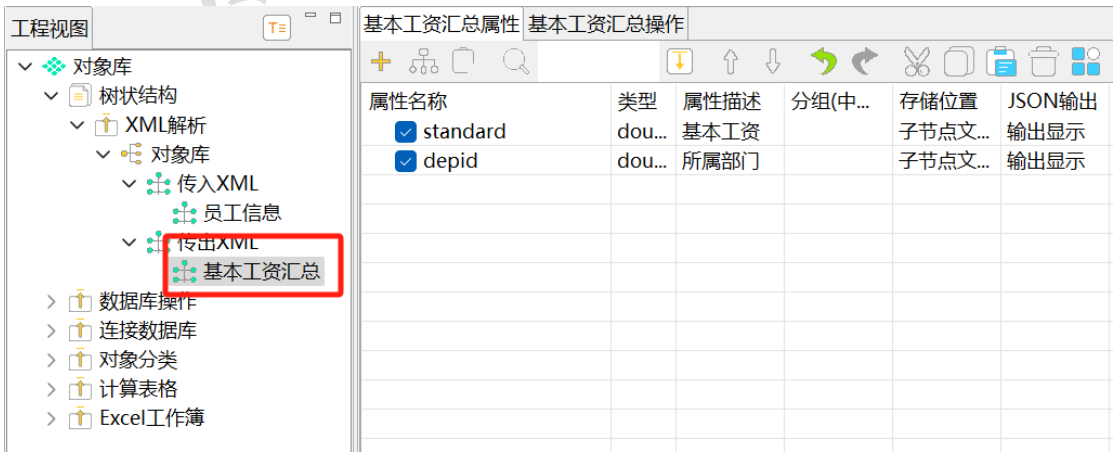
在“员工信息”中添加如下字段信息：



在“传出 xml”的操作中添加方法，步骤同上：



最后在“基本工资汇总”中添加如下字段：

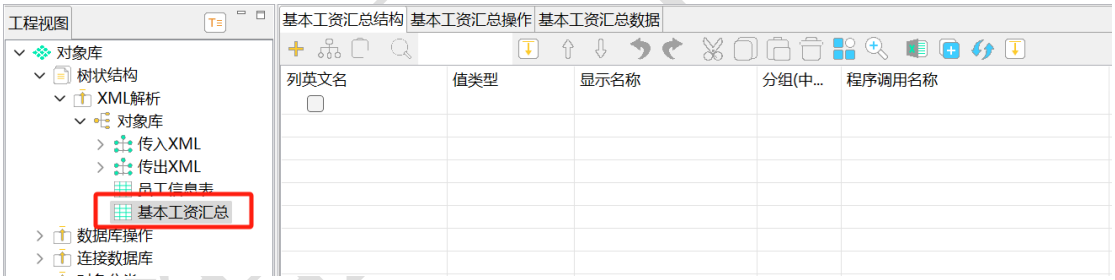


4．添加计算表格

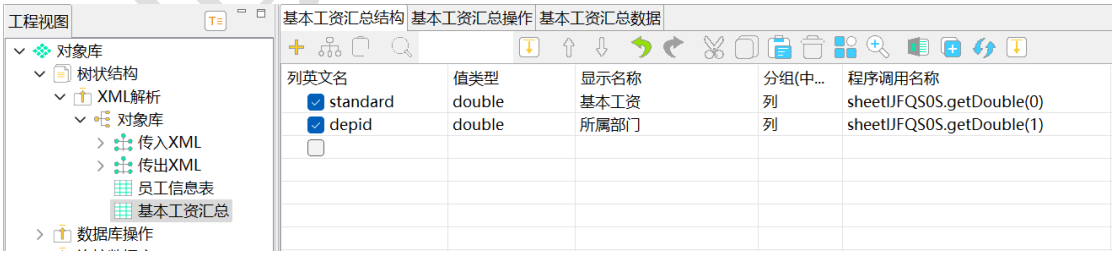
右键对象库，添加第一个计算表格，命名为“员工信息表”：

在“员工信息表”中添加如下表字段：

右键对象库，添加第二个计算表格，命名为“基本工资汇总”：



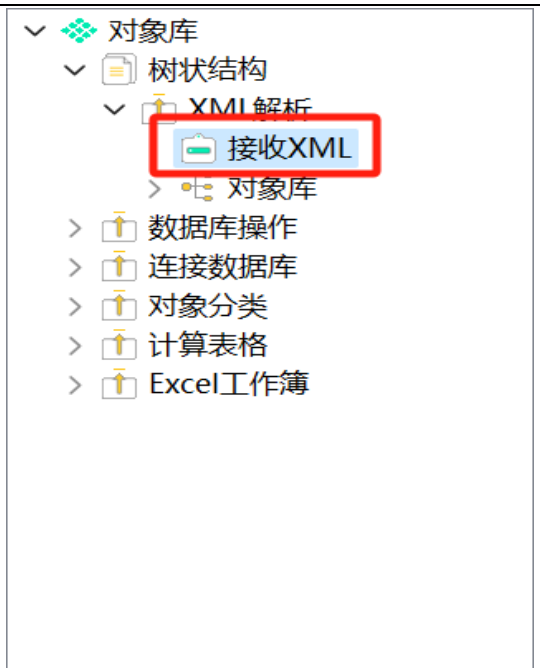
并添加表字段：



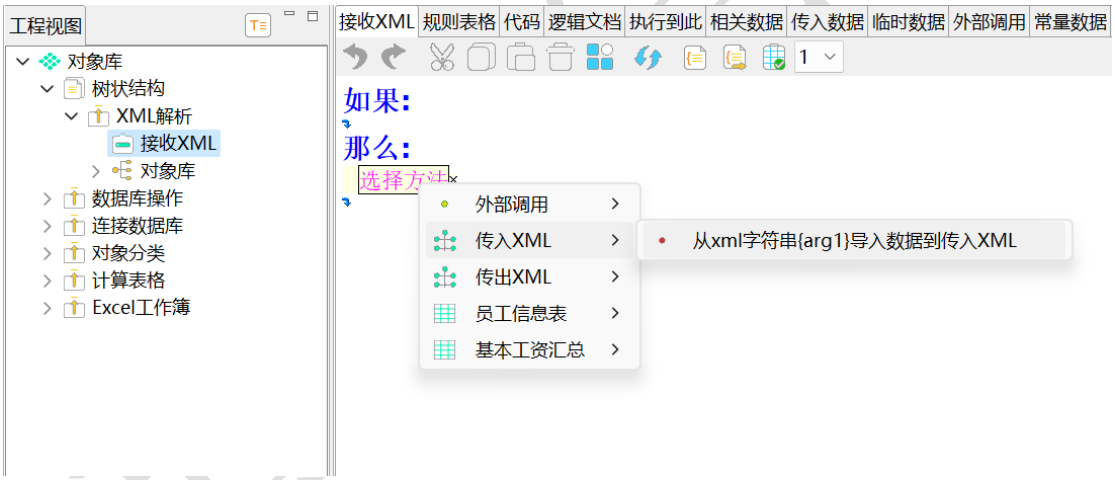
5．新建规则

(1) 创建“接收 xml”规则

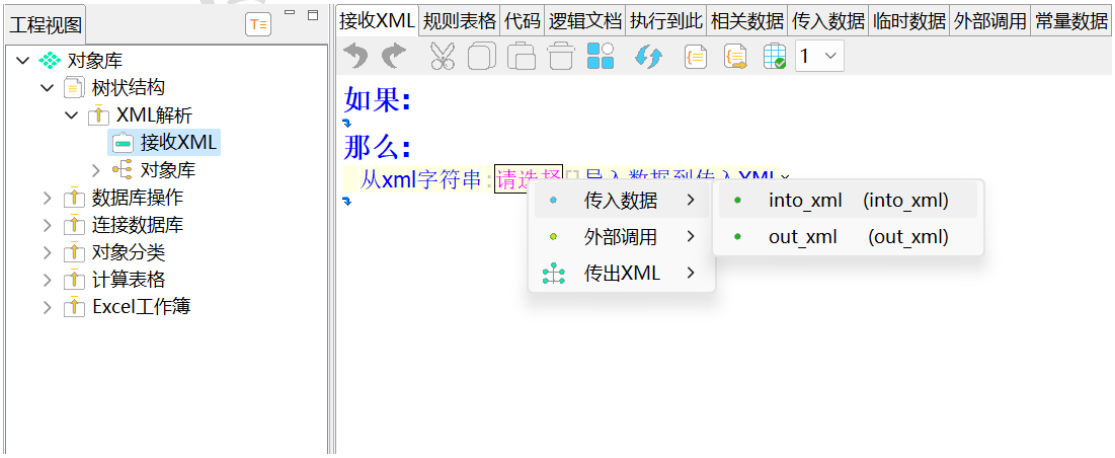
新建规则“接收 xml”，用于接收 xml 数据，并解析到“传入 xml”的树状结构中，点击“XML 解析”规则包，添加规则，命名为“接收 xml”：



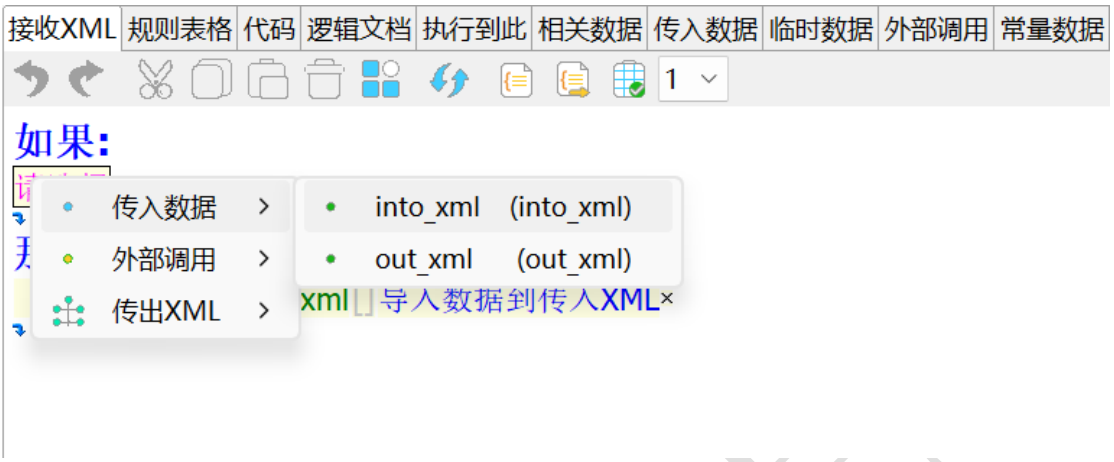
在“接收 xml”规则中，点击添加动作，执行方法→传入 xml→从 xml 字符串{arg1}中导入数据到传入 xml。



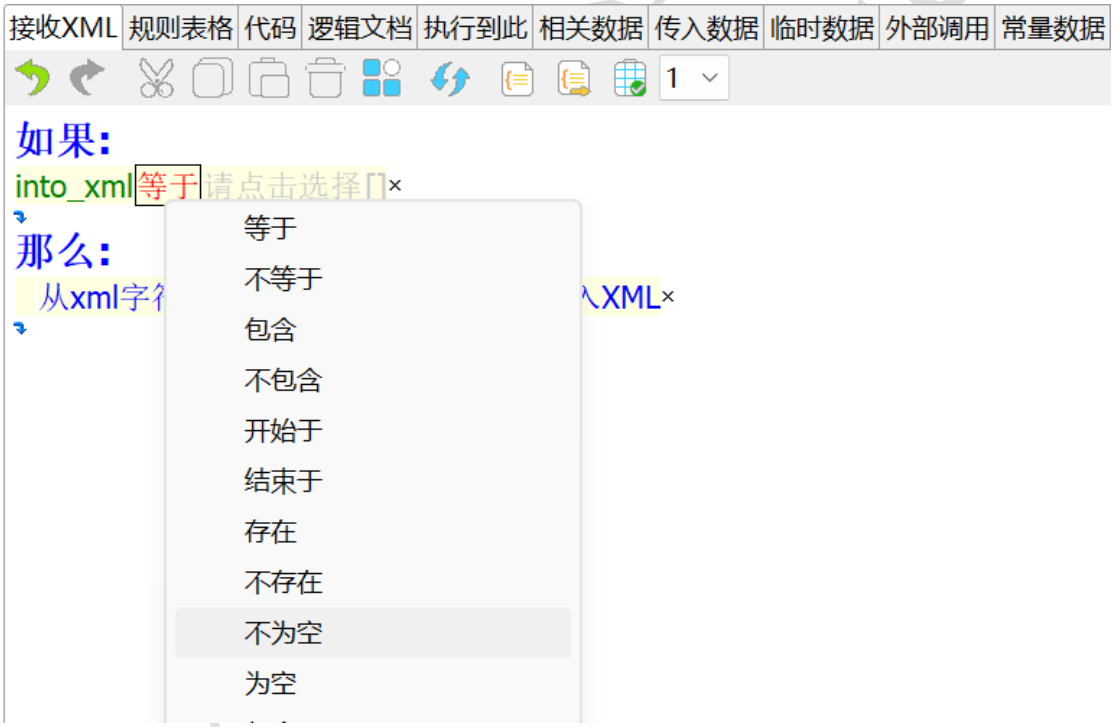
点击“点击请选择[]”选择 选择值→传入数据→传入 xml



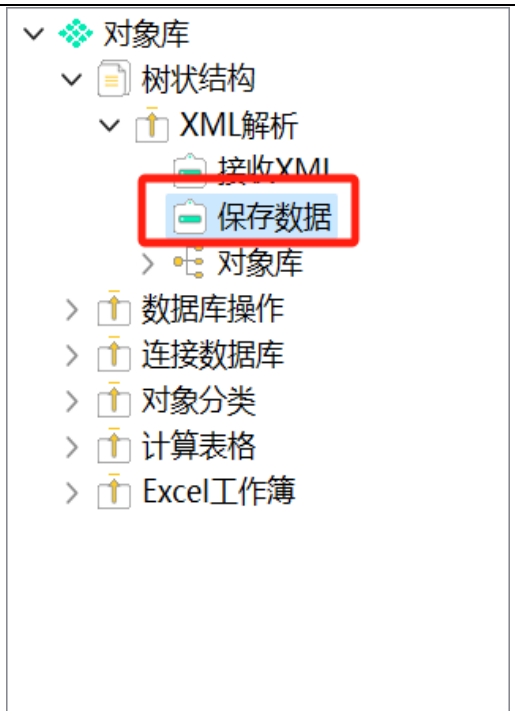
在此之前我们需要对“传入 xml”进行判断，如果其值不为空，则才能执行下面的操作.
点击“如果”下面的“ ”添加动作，选择 传入数据→传入 xml：



点击“等于”在弹出菜单中选择“不为空”：

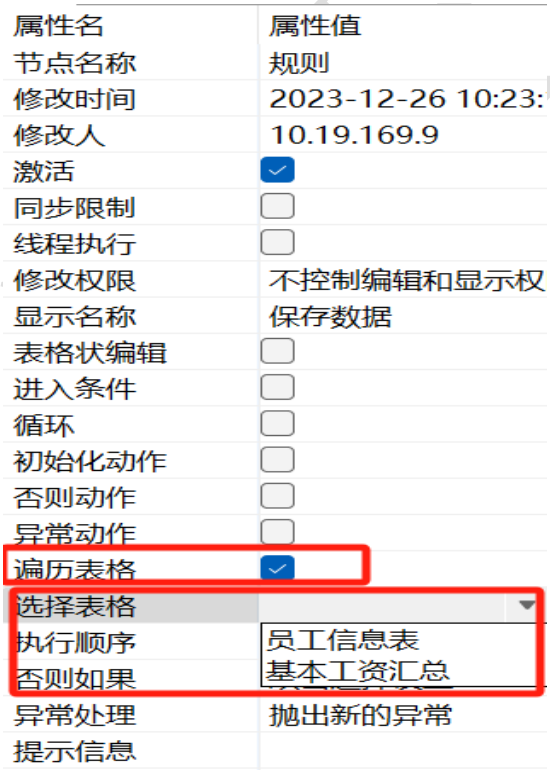


(2) 创建“保存数据”规则
右键“XML 解析”规则包，添加规则，命名为“保存数据”：

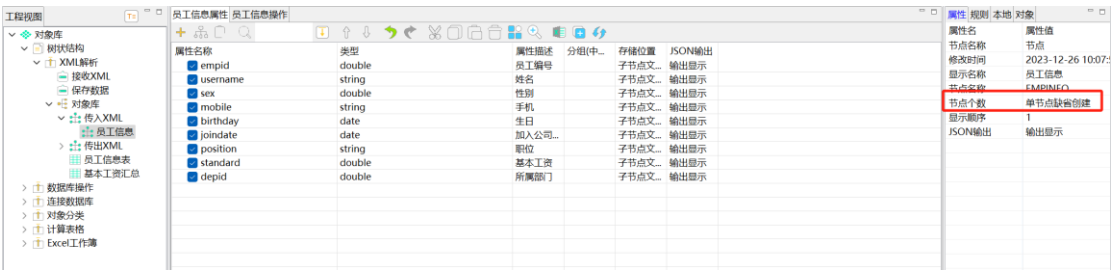


要使“员工信息”中的数据全部保存到“员工信息表”，这需要对“员工信息”进行遍历，从而逐条添加到内存表。

点击“保存数据”规则，在右侧属性窗口将“遍历表格”勾选，下方出现一行“选择表格”，这里我们需要选择“传入 xml.员工信息”，但点击下拉列表里面并没有这个选项，是因为“员工信息”的节点个数默认为“单节点缺省创建”，应将其改为“多节点可以为空”：

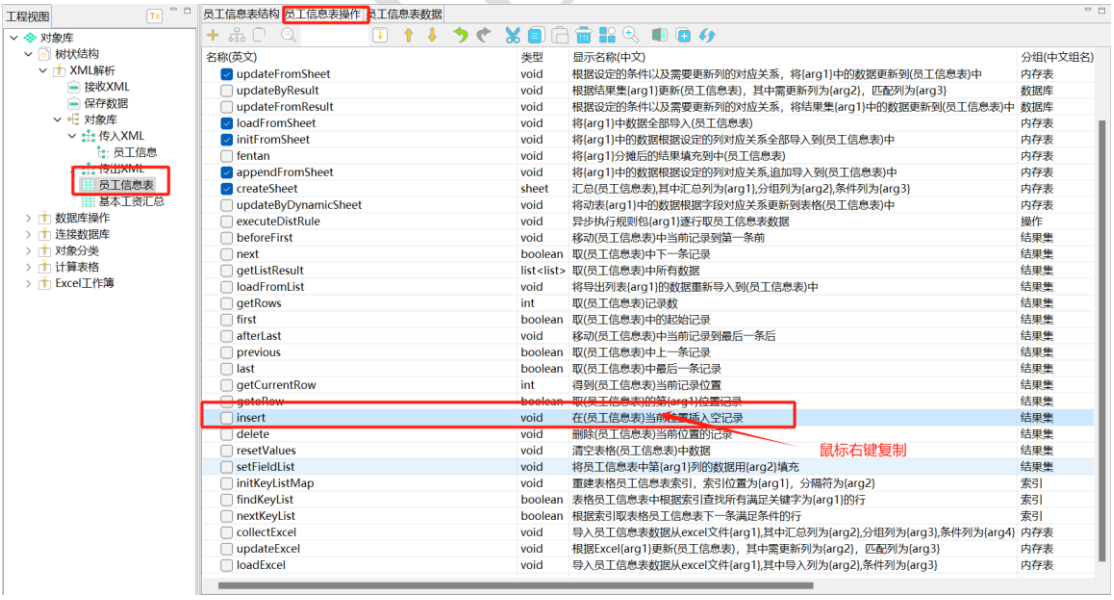


点击“员工信息”树状结构，在右侧属性窗口将“节点个数”的属性值通过点击下拉列表由“单节点缺省创建”更换为“多节点可以为空”：

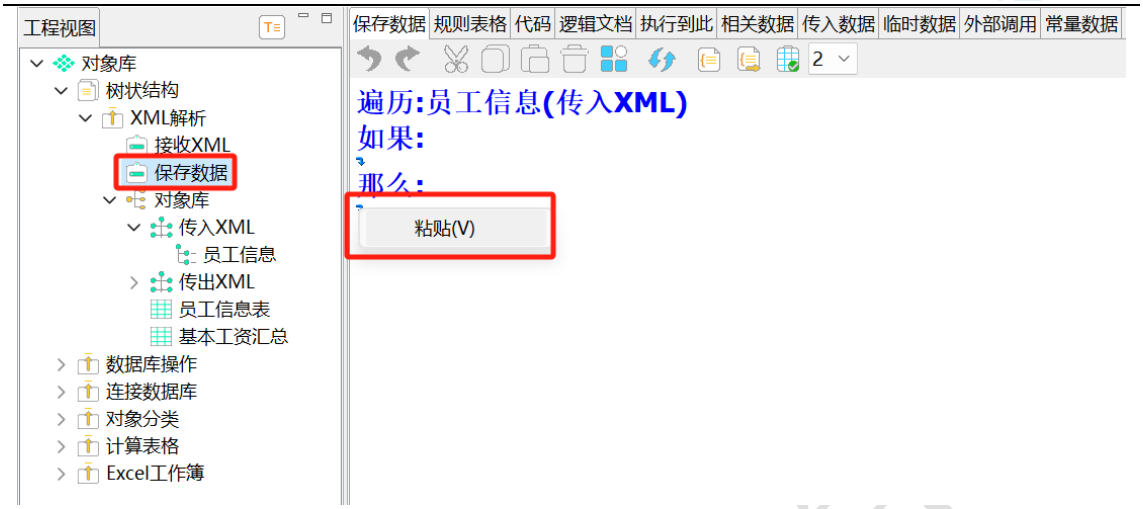


回到“保存数据”规则，在属性窗口中选择表格“传入 xml.员工信息”：

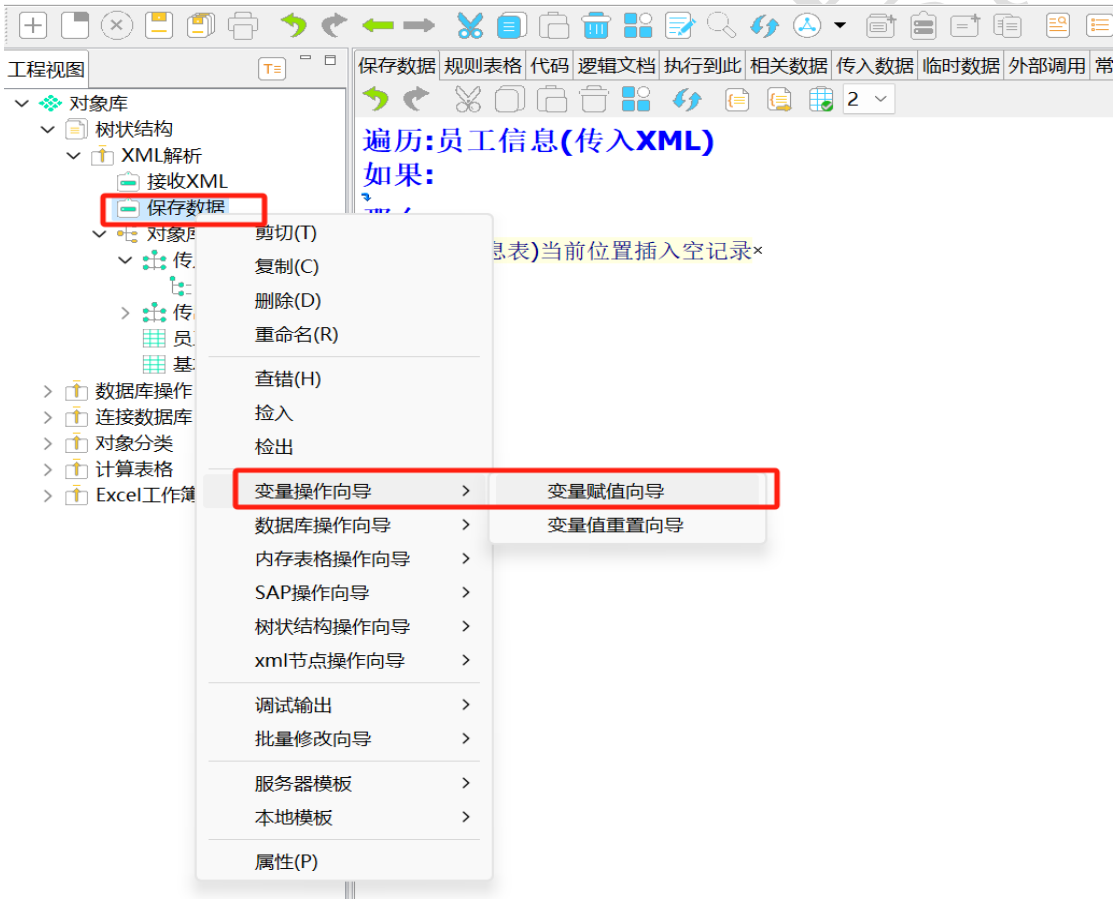
要将一个表中的数据保存到另一张表中，首先需要有插入动作。在“员工信息表”的“员工信息表操作”中，选择 insert 插入方法复制：



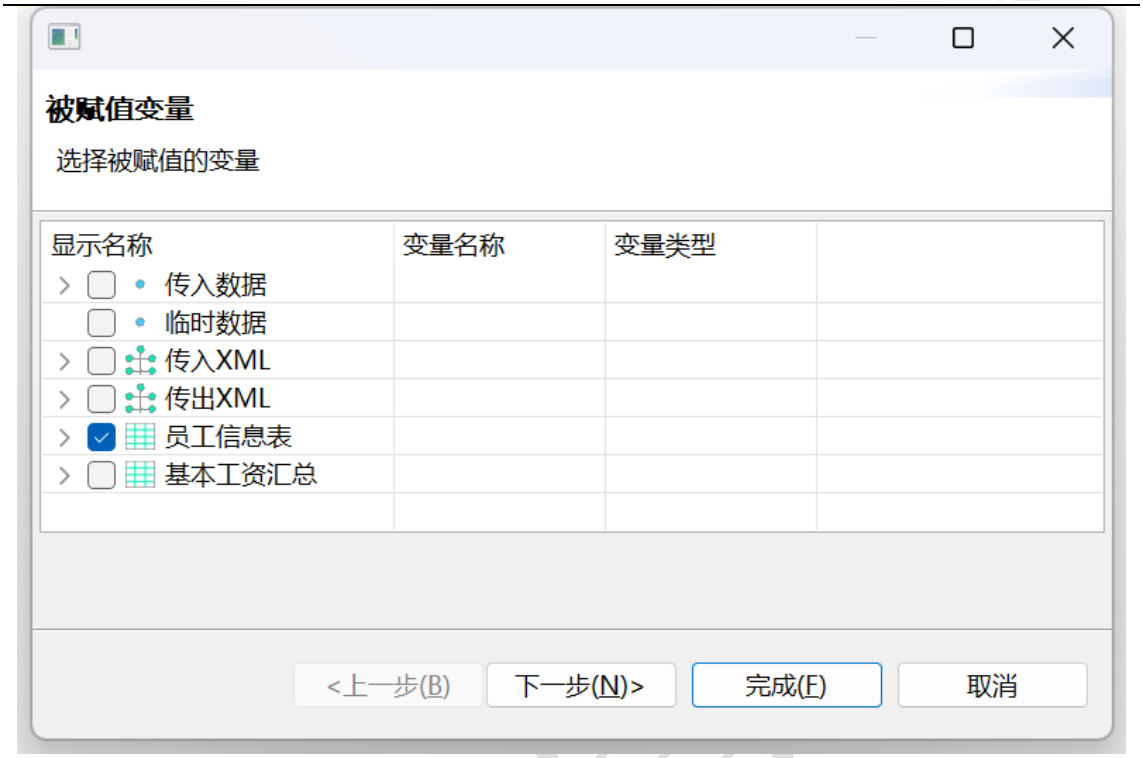
在“保存数据”规则的编辑窗口中，右键添加动作，选择粘贴：



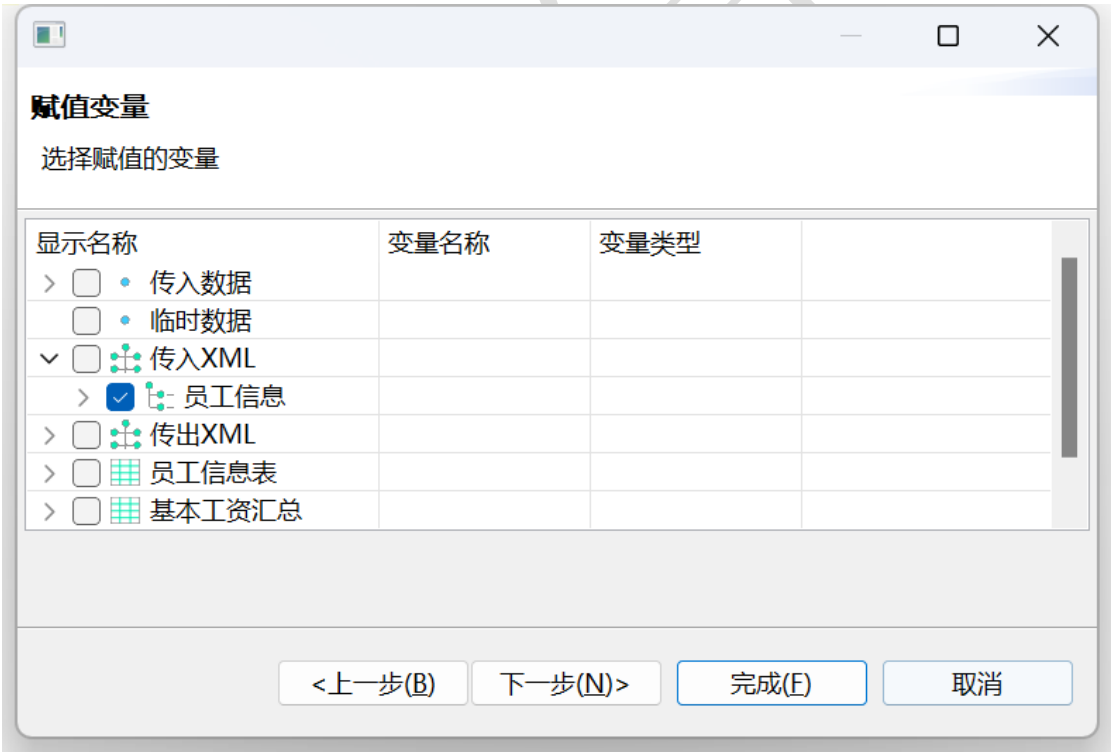
右键“保存数据”规则，选择“变量操作向导”下的“变量赋值向导”：



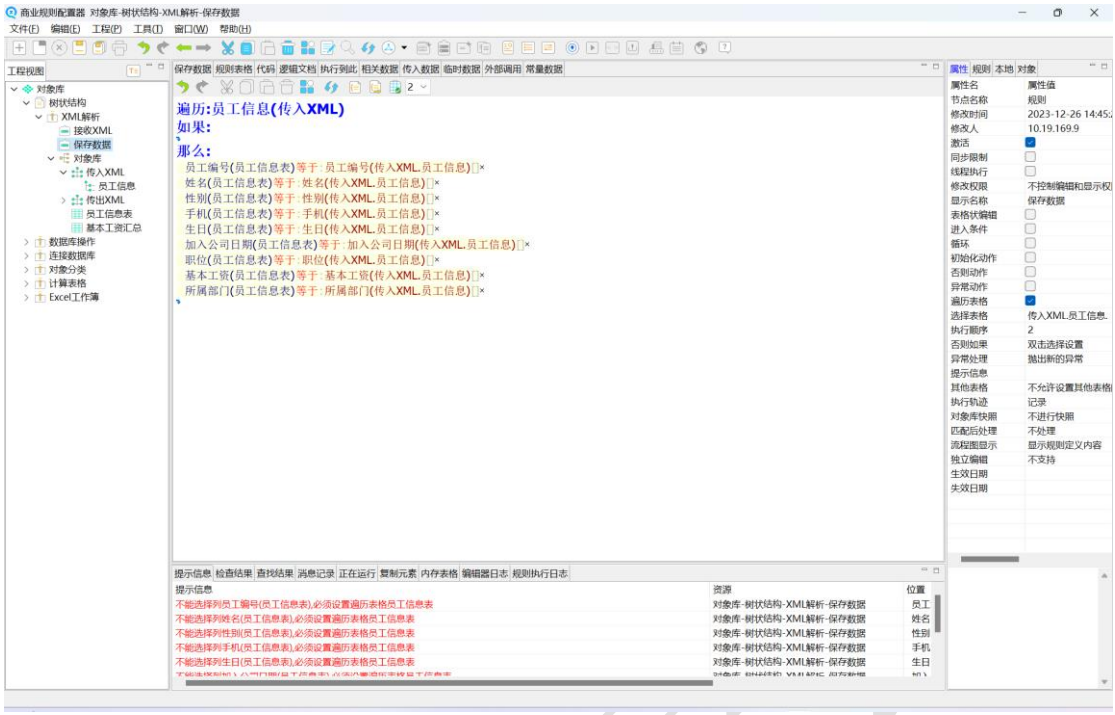
勾选“员工信息表”，点击下一步：



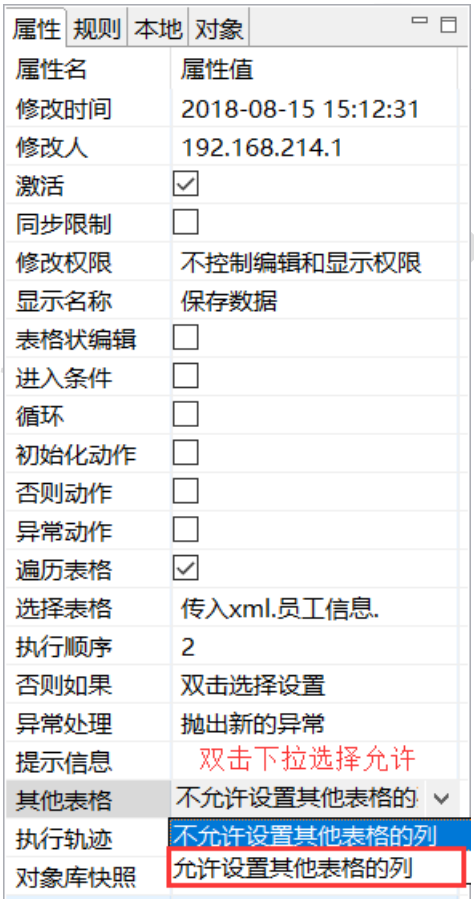
勾选“传入 xml”下的“员工信息”，点击完成：



点击完成之后，自动生成给表字段赋值操作，这时在下方消息窗口中会出现红色字体报错，提示没有遍历员工信息表：



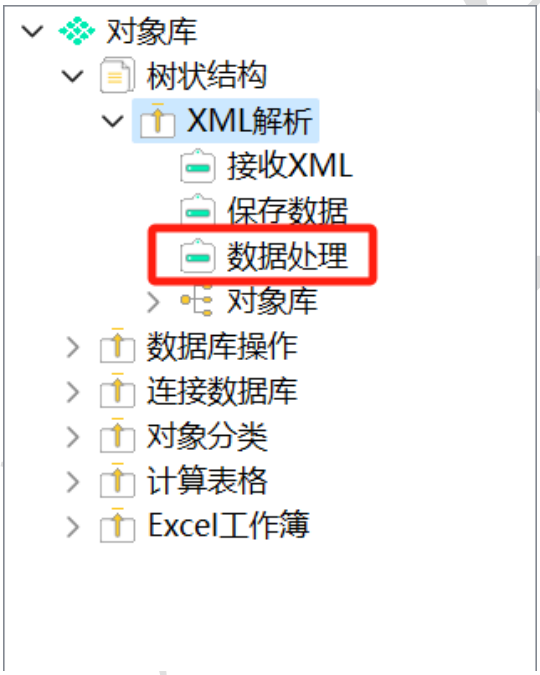
在“保存数据”规则的右侧属性编辑窗口中将“其他表格”的属性值由“不允许设置其他表格的列”更改为“允许设置其他表格的列”：



更改好后，点击编辑窗口工具栏中的刷新按钮，下方消息窗口的错误提示消失：

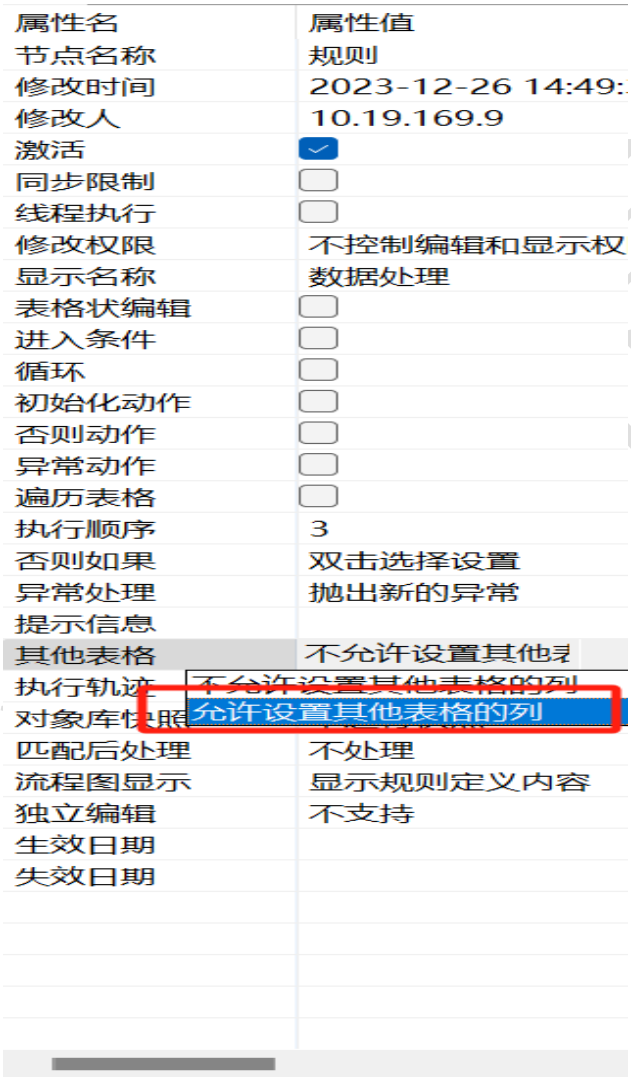
(3) 创建“数据处理”规则

点击“XML 解析”规则包，右键添加规则，并命名为“数据处理”：

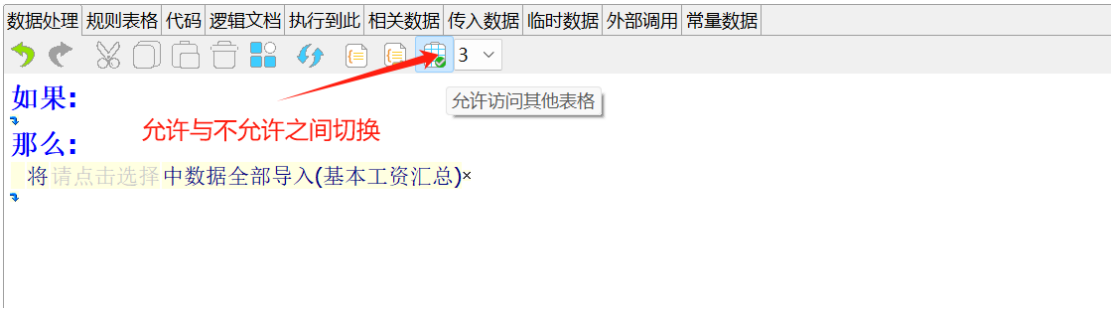


点击添加动作，执行方法→选择方法→基本工资汇总→内存表→将{arg}中数据全部导入（基本工资汇总）

在属性编辑窗口中，将“其他表格”的属性值改为“允许设置其他表格的列”：

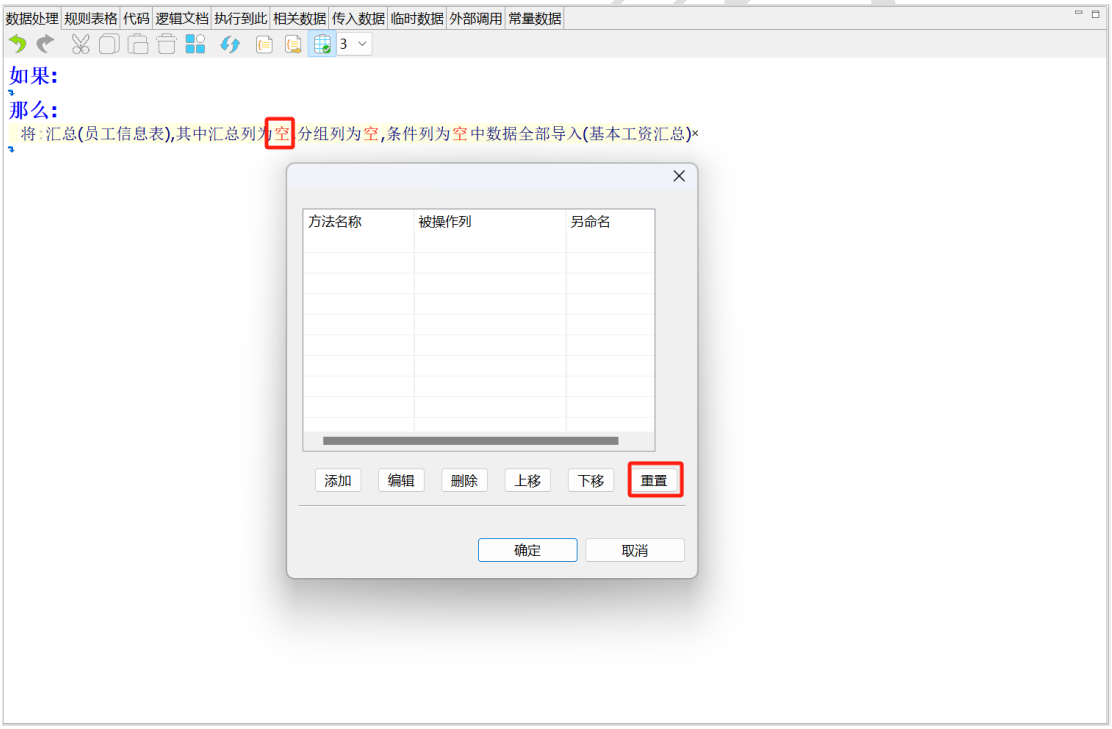


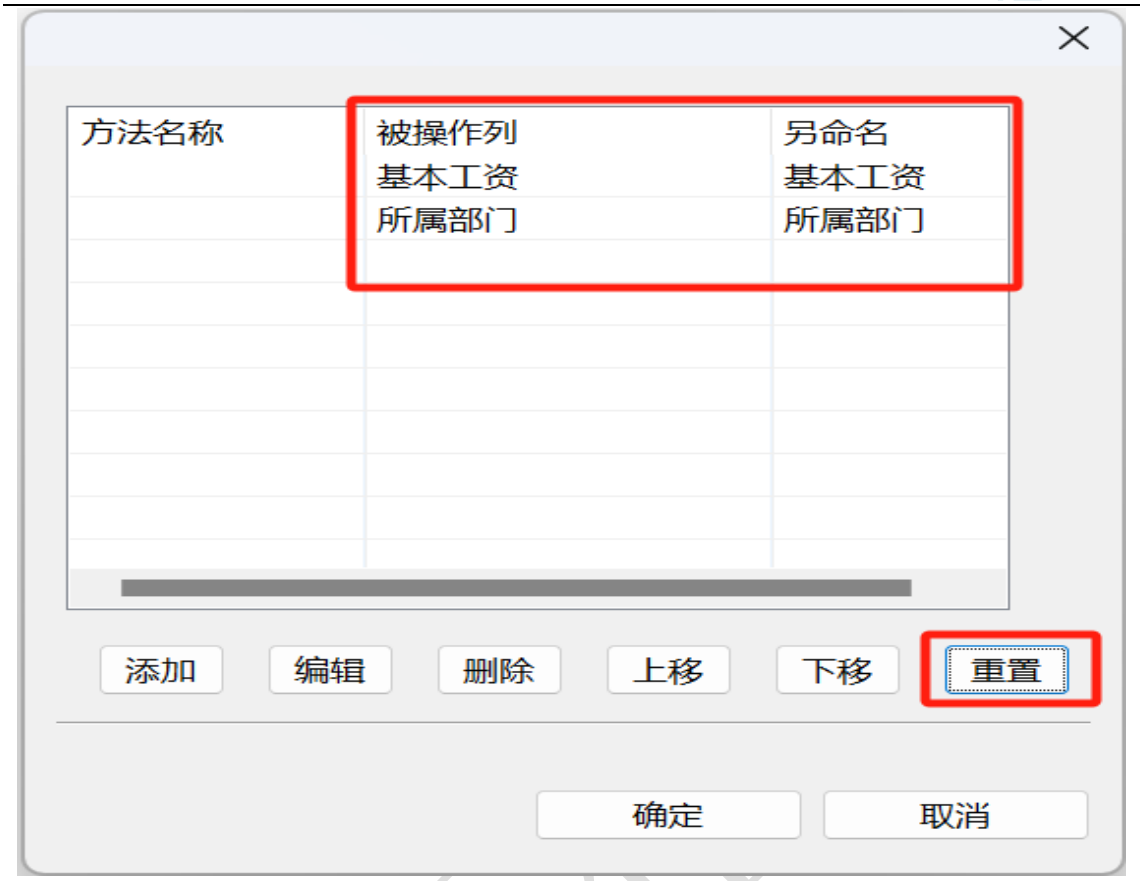
或者在编辑窗口，点击工具栏上的“ ”按钮，设置为“允许设置其他表格的列”：



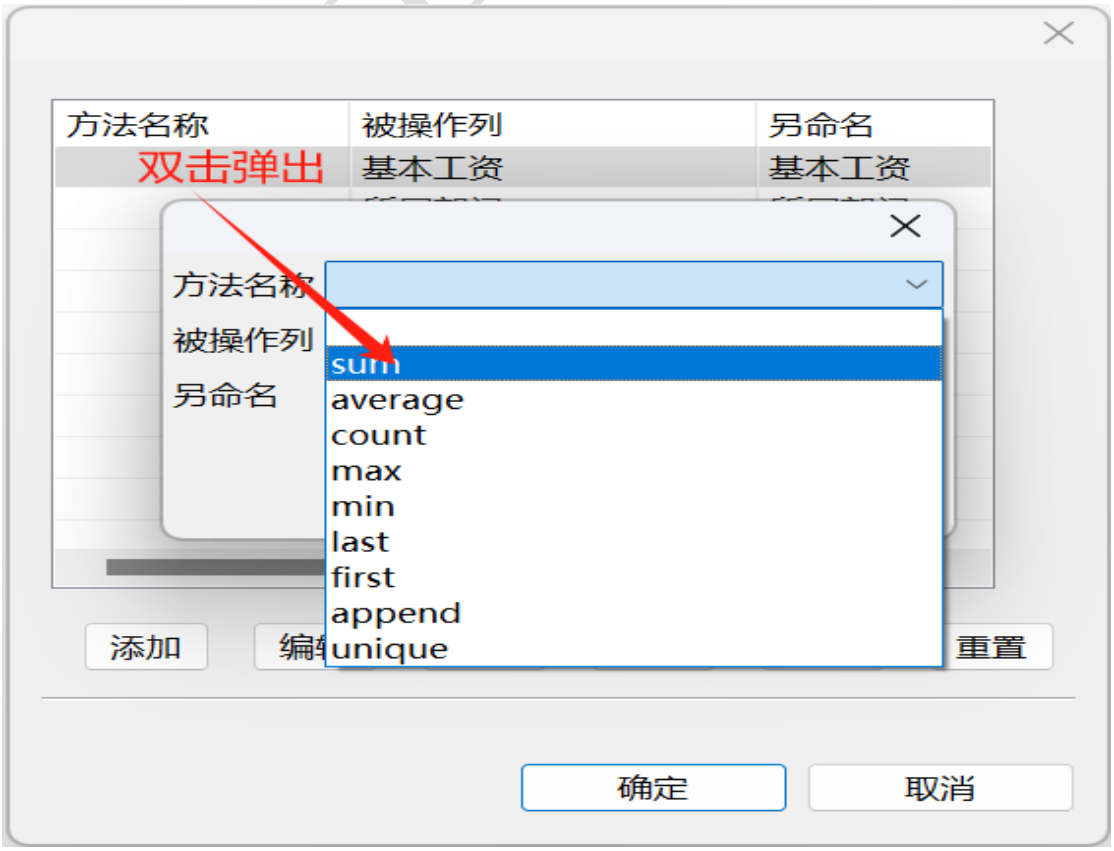
点击“点击请选择[]”选择“选择值”→员工信息表下的方法，如图：

点击第一个空，选择汇总的列，在弹出框中点击重置按钮，自动添加所匹配到的列：

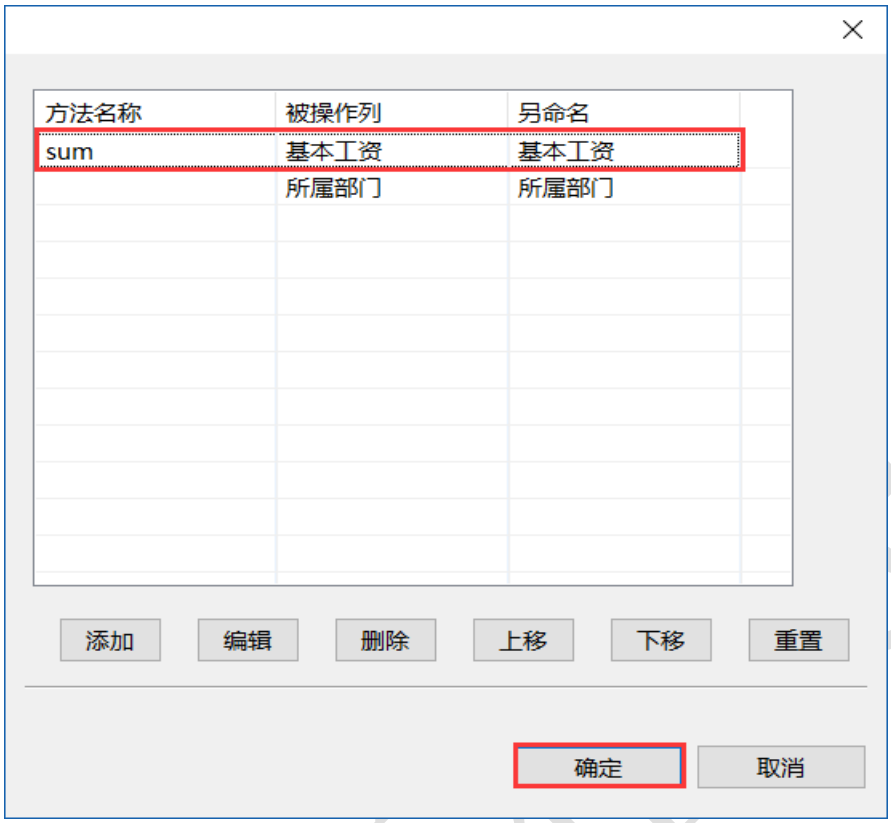




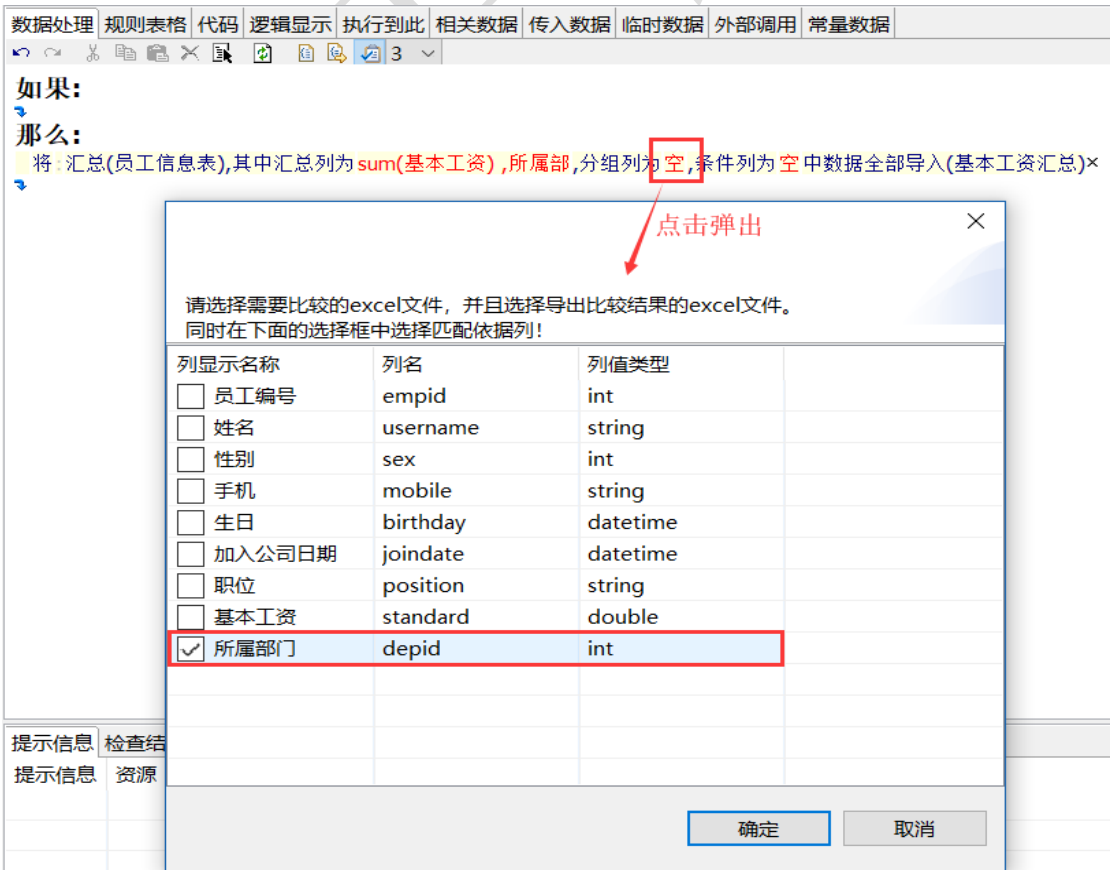
双击“基本工资”这一行，在弹出框中选择“方法名称”点击下拉，选择“sum”方法：



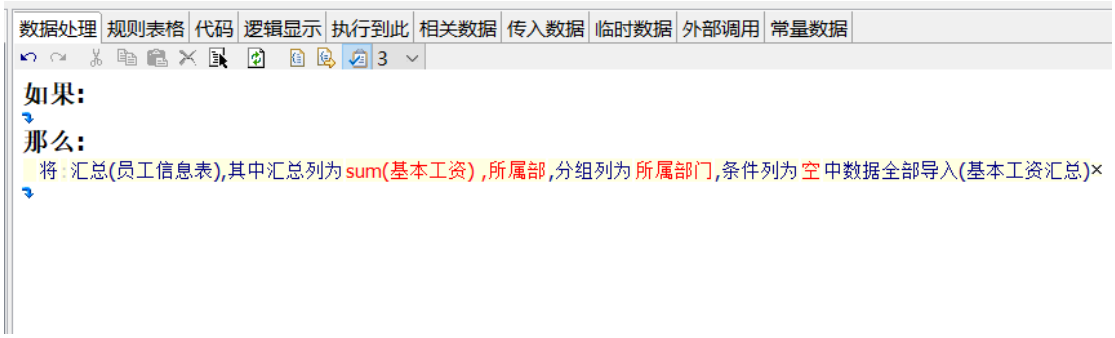
完成后点击确定，基本工资方法添加完成：



点击第二个空，添加分组列，在弹出框中将“所属部门”勾选：

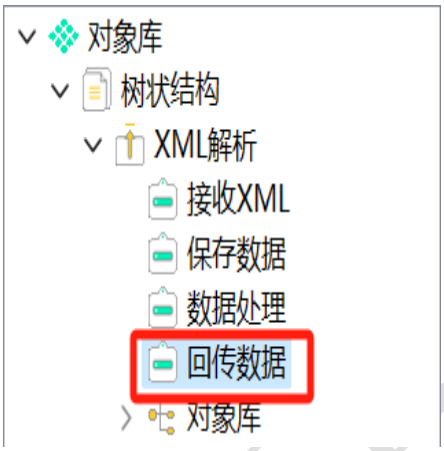


完成后规则如下：

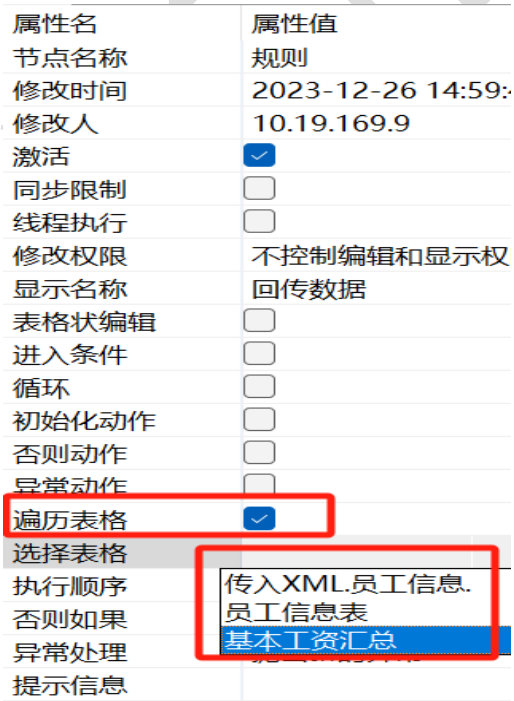


(4) 创建“回传数据”规则

右键“树状结构”规则包，添加规则，并命名为“回传数据”：

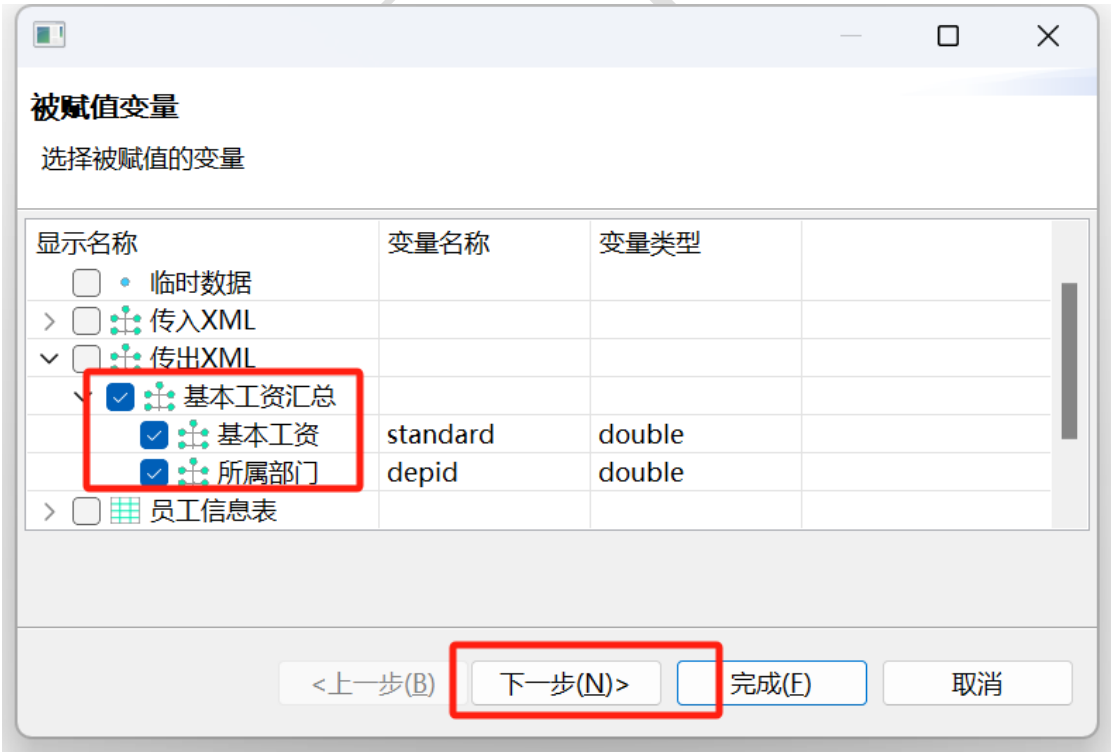


遍历“基本工资汇总”表，在“回传数据”规则的属性窗口设置遍历表格：



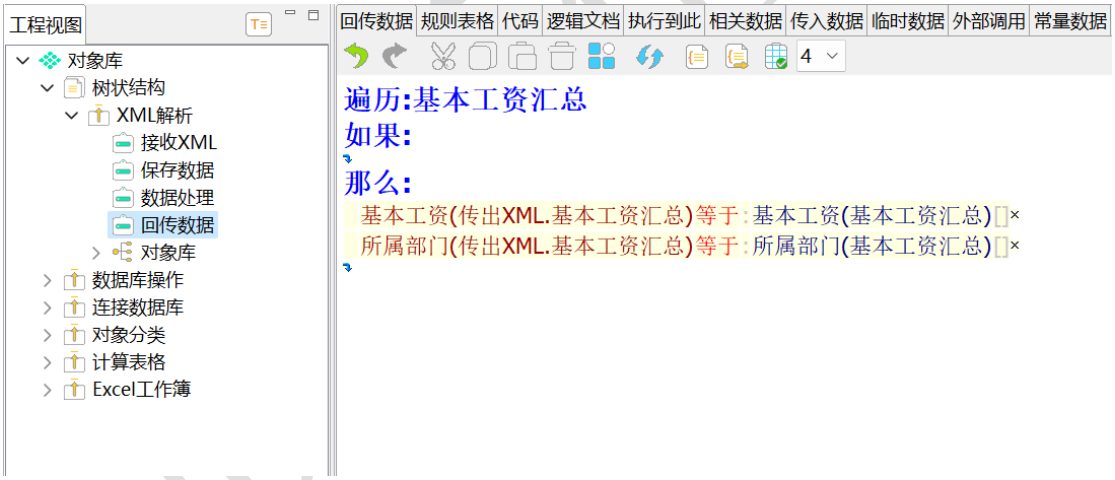
右键“回传数据”规则，选择 变量操作向导→变量赋值向导：

勾选“传出 xml”下的“基本工资汇总”，点击下一步：



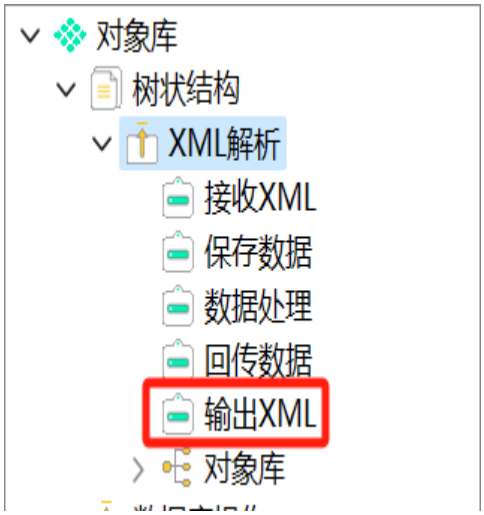
勾选“基本工资汇总”，点击完成。

完成之后如下图：

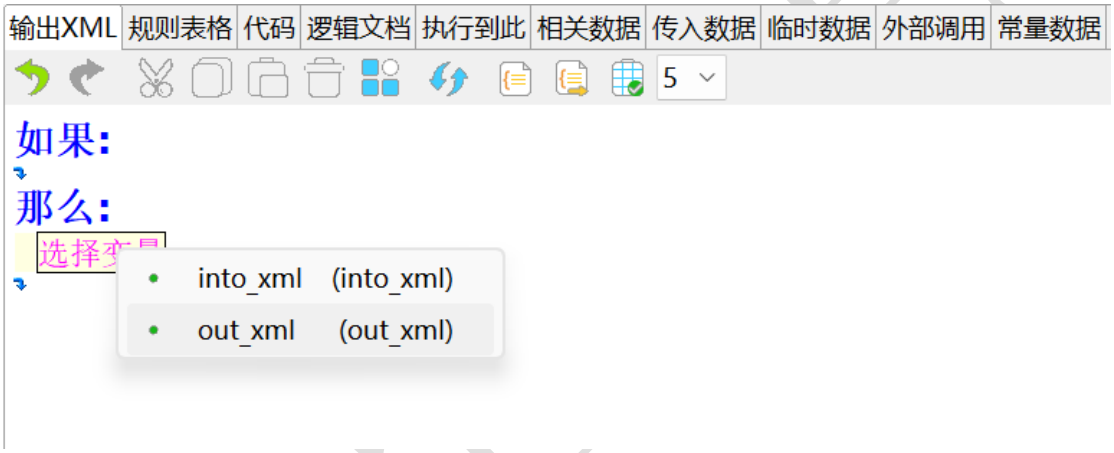


(5) 创建“输出 xml”规则

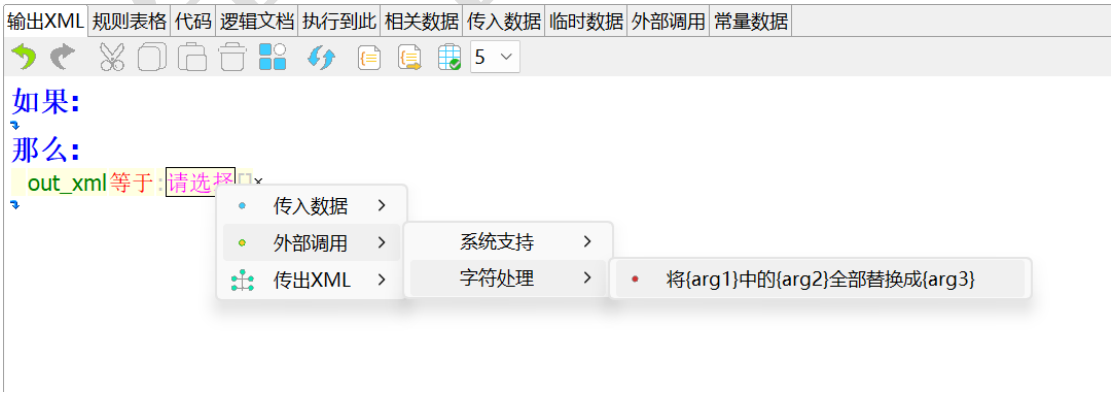
点击“XML 解析”规则包，右键添加规则，命名为“输出 xml”：



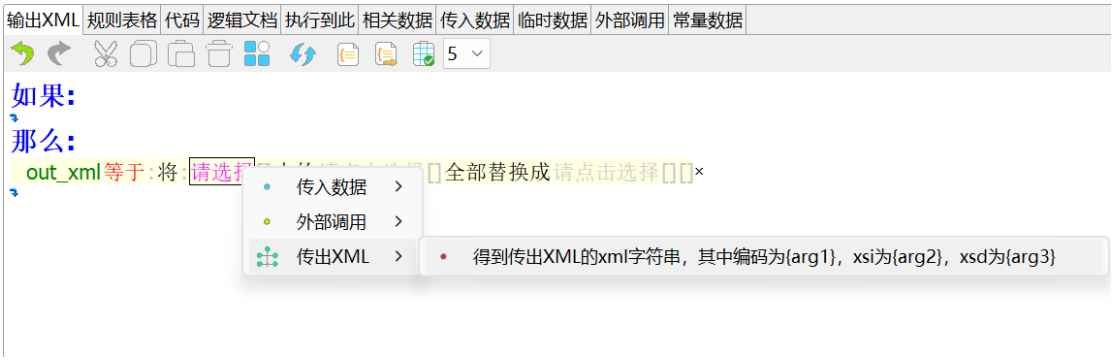
点击添加动作，变量赋值→传出 xml：



点击“点击请选择[]”，选择值→外部调用→字符处理：

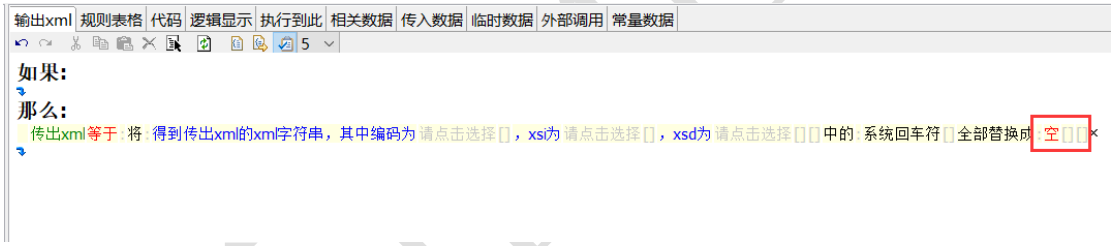


点击第一个“ 请点击选择[] ”，选择 选择值→传出 xml：

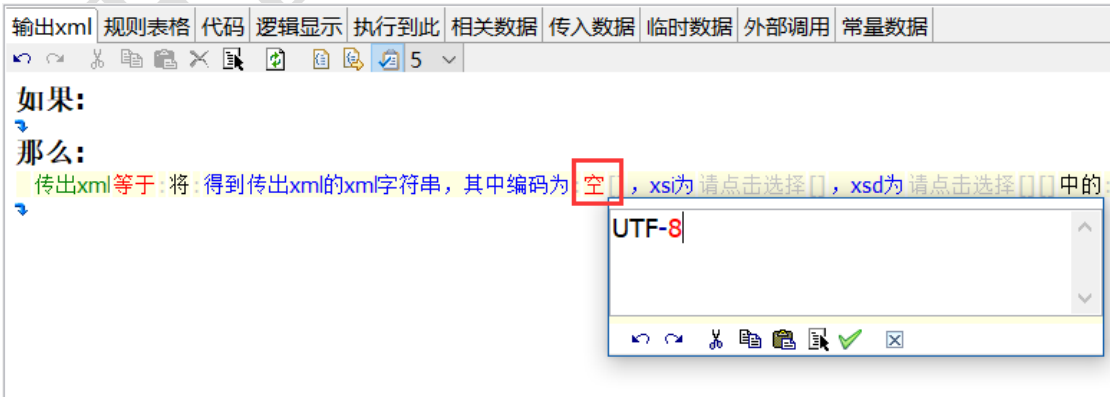


点击第二个“请点击选择[]”，选择 选择值→外部调用→系统支持：

点击第三个“请点击选择[]”，选择输入常量值，值为空：

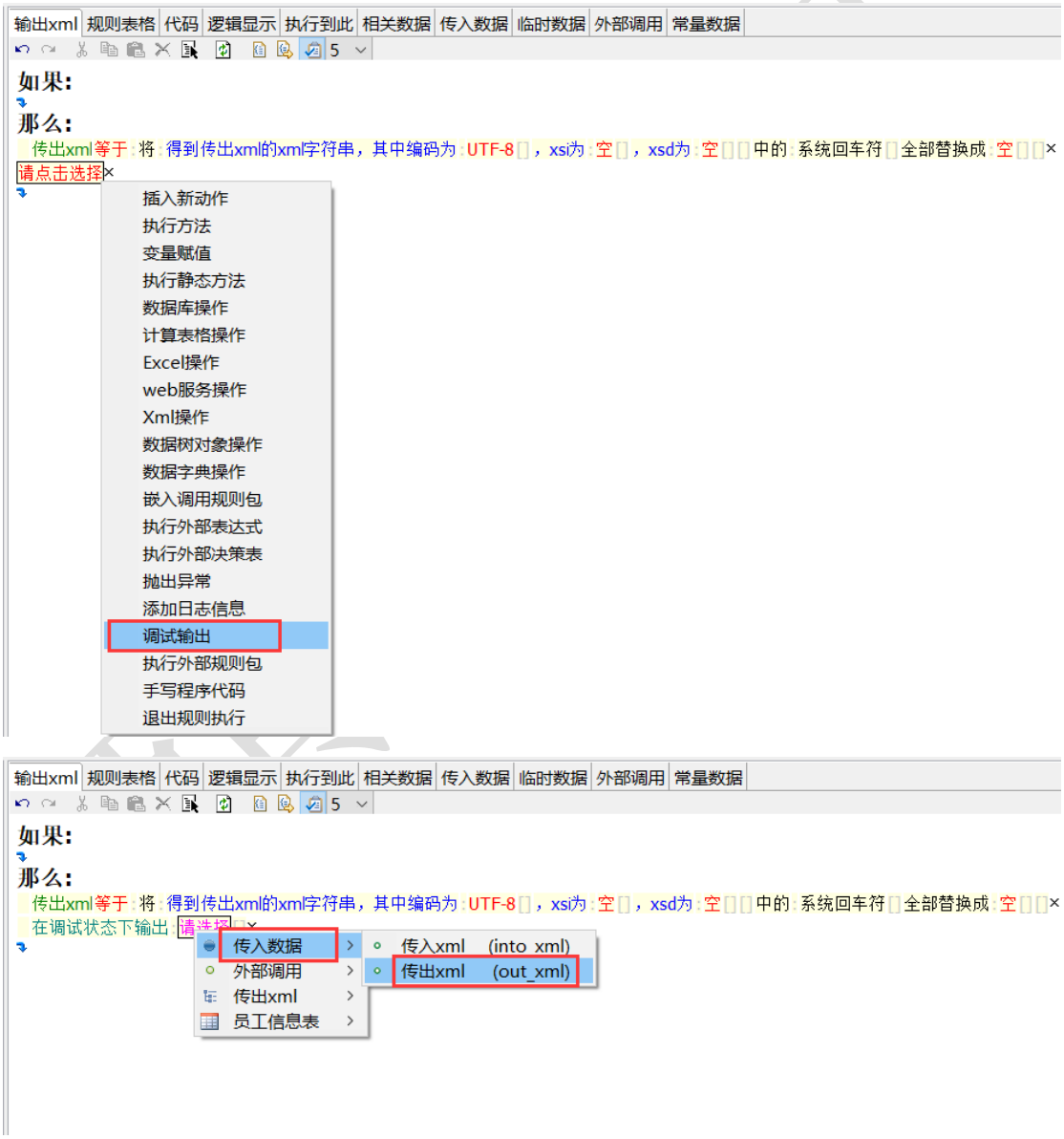


在“得到传出 xml 的 xml 字符串……”中，点击第一个“请点击选择[]”选择输入常量值，编码设置为“UTF-8”：



在“得到传出 xml 的 xml 字符串……”中，依次将第二第三个“请点击选择[]”通过点击选择输入常量值设置为空，如下图：

点击添加动作，将“传出 xml”输出在控制台消息窗口，选择“调试输出”（等同于 System.out.println()）。



规则编写完成后，点击“”保存并编译。

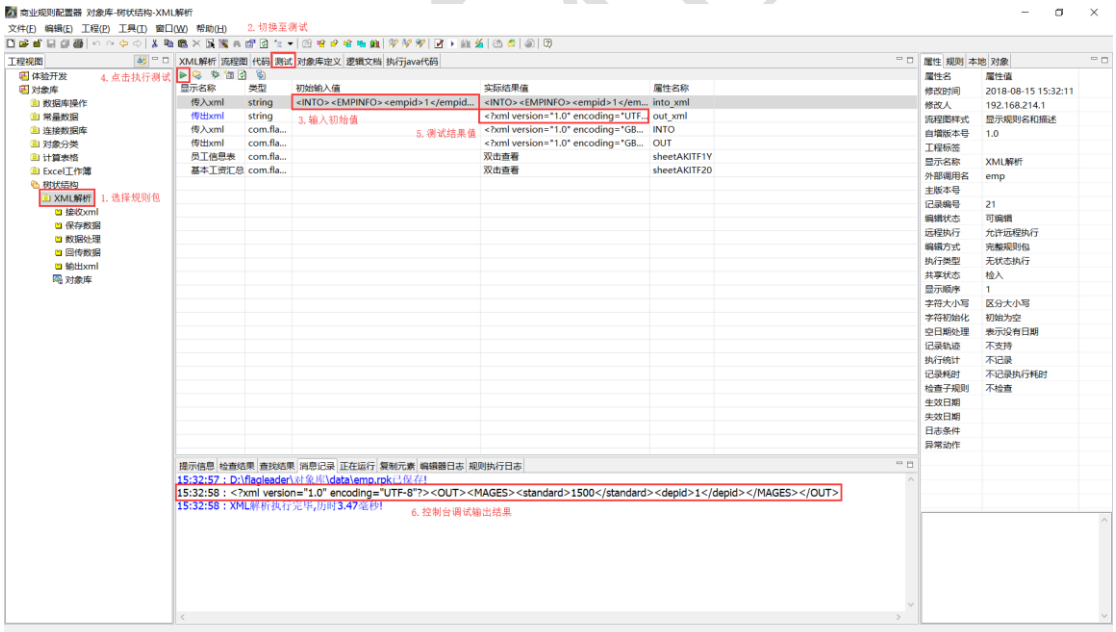
6. 测试

点击“XML 解析”规则包，在编辑窗口中输入初始值进行测试。

这里的初始值是 xml 字符串，因此要测试首先需要编写 xml 格式的字符串，如下所示：

```
<INTO>
    <EMPINFO>
        <empid>1</empid>
        <username>小张</username>
        <sex>0</sex>
        <mobile>123456789</mobile>
        <birthday>1979-06-08 00:00:00</birthday>
        <joindate>2009-04-01 00:00:00</joindate>
        <position>业务员</position>
        <standard>1500</standard>
        <depid>1</depid>
    </EMPINFO>
</INTO>
```

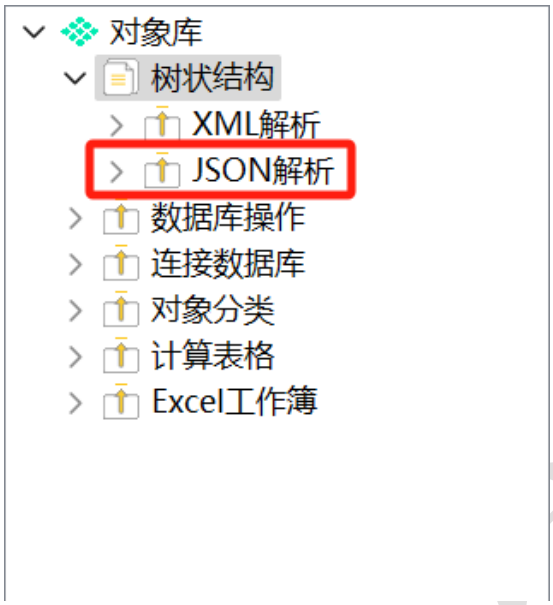
其中<INTO>标签是“传入 xml”的节点名称，<EMPINFO>标签是“员工信息”的节点名称，<EMPINFO>标签中的<empid>、<username>等是“员工信息”树状结构中的属性，而我们要传入的值就放在属性标签的包裹中。编写完成后需要对其进行格式化，去掉所有的空格和换行，使之变成一行字符串，然后将这行字符串复制到测试中的“初始输入值”一栏中，点击开始测试：



2.3.5.2. JSON 解析

1. 创建规则包

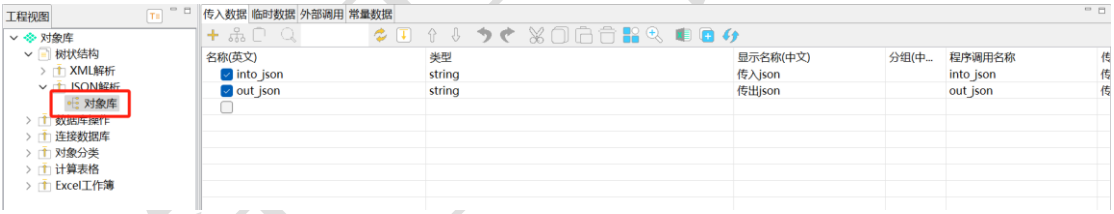
在规则组“树状结构”中，新建规则包，并将其命名为“JSON 解析”：



2. 创建对象库变量

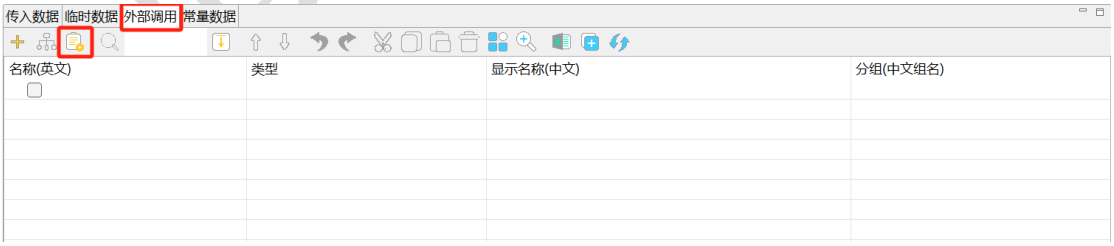
(3) 添加传入数据

打开“JSON 解析”规则包，在对象库传入数据中添加如下变量：

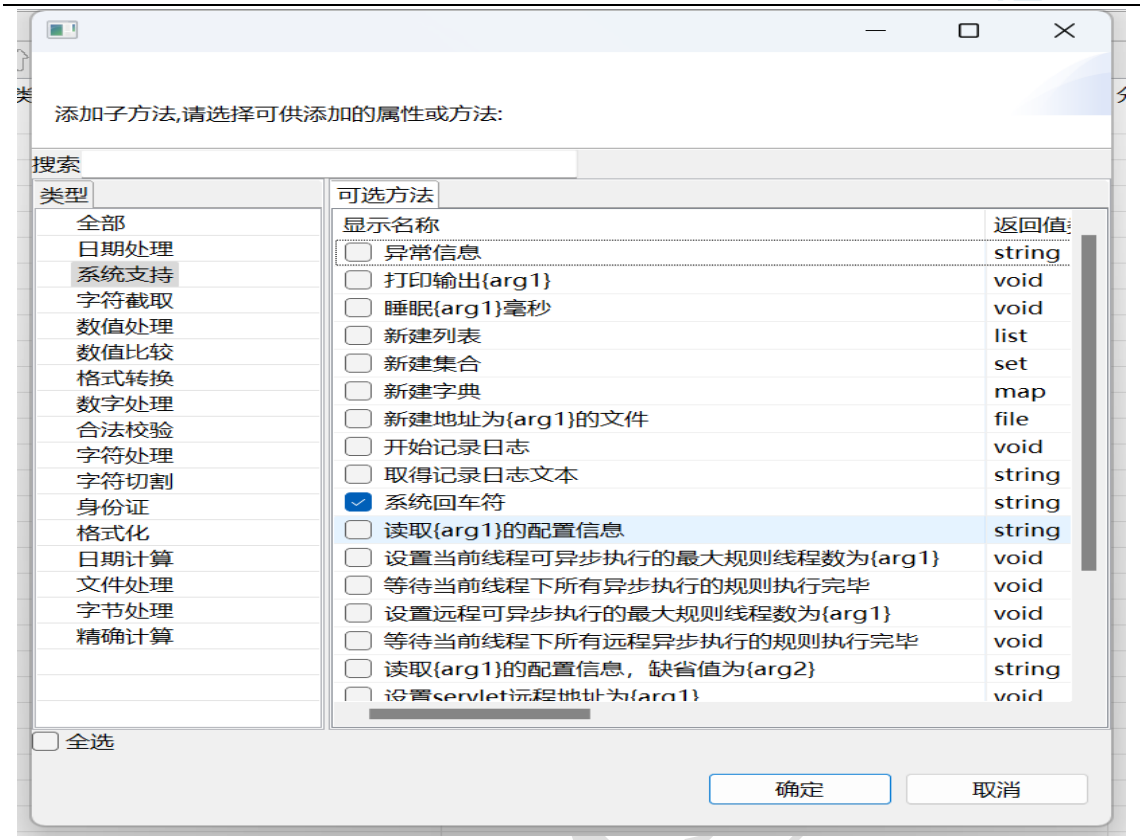


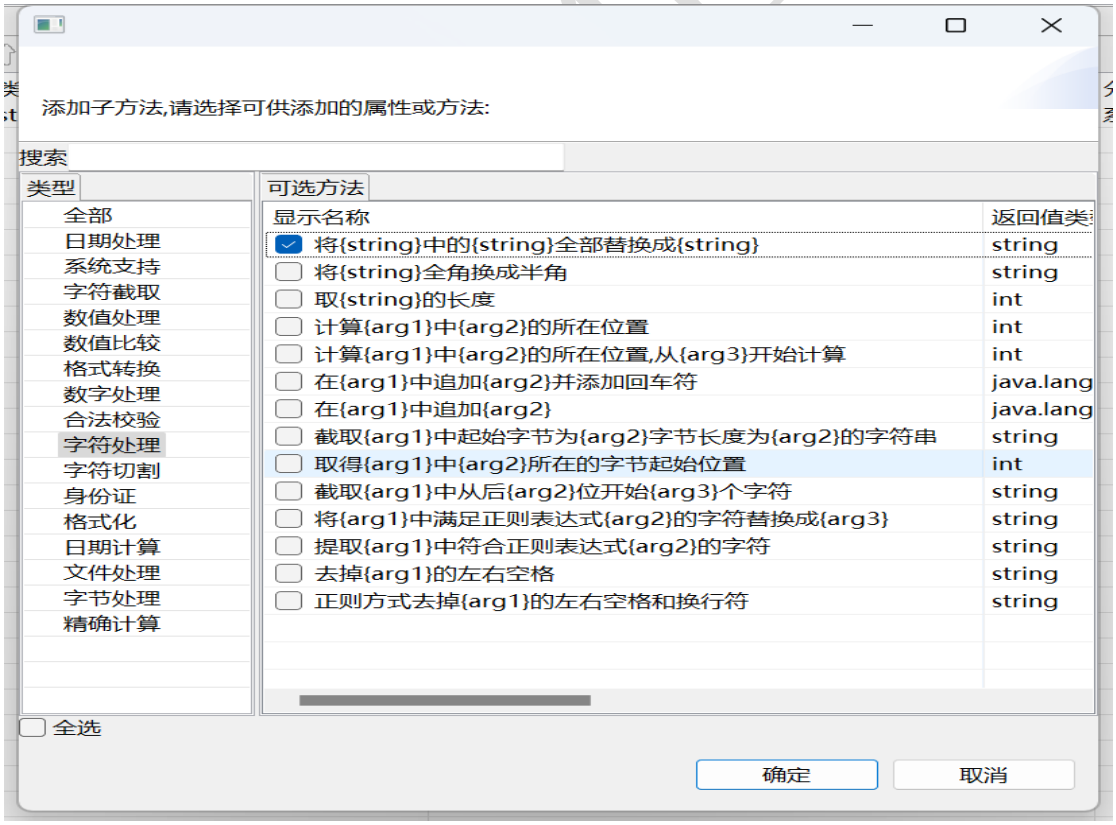
(4) 添加外部调用

在外部调用中点击“ ”添加公式：



分别添加“系统支持”中的“系统回车符”方法，和“字符处理”中的“将{arg1}中的{arg2}全部替换为{arg3}”





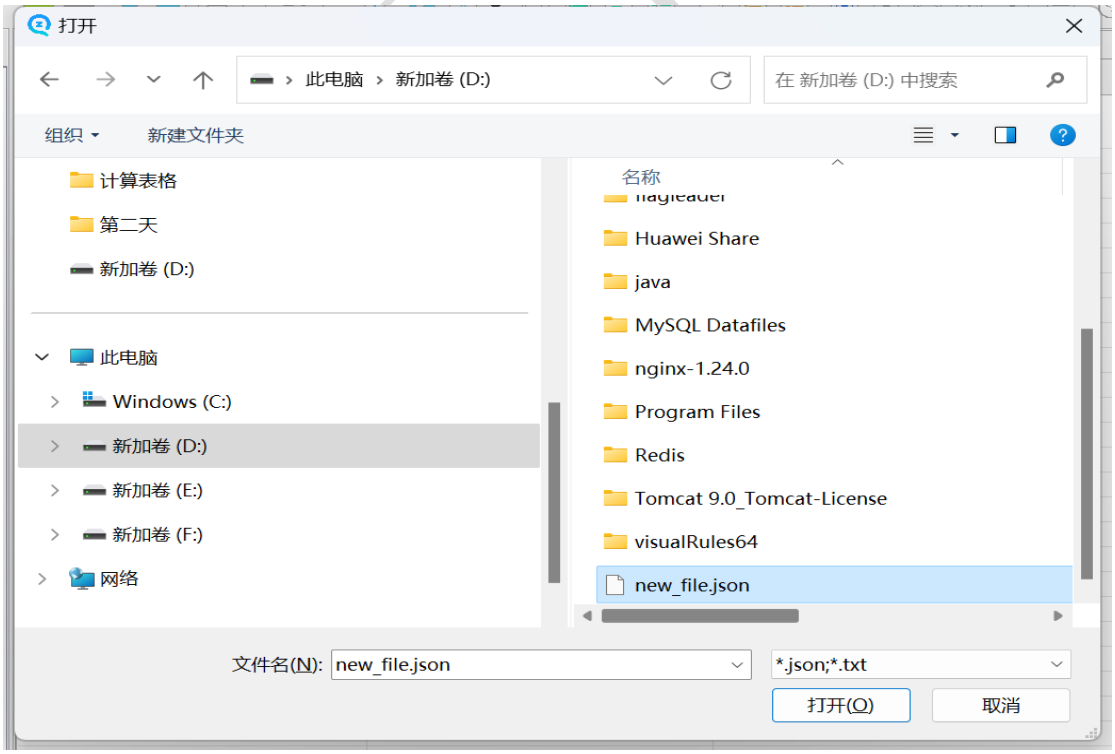
3. 根据 JSON 生成树结构

右键对象库，选择“根据 JSON 生成树结构”，JSON 文件如下：

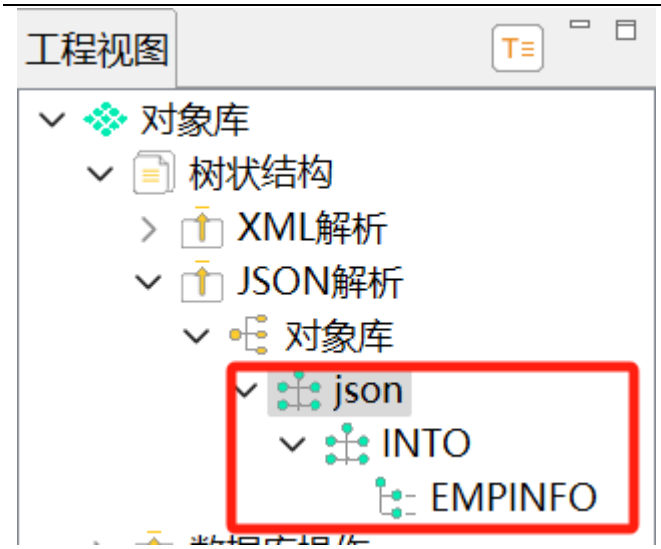
```
{
```

```
    "INTO": {
        "EMPINFO": [
            {
                "empid": "1",
                "username": "小张",
                "sex": "0",
                "mobile": "123456789",
                "birthday": "1979-06-08 00:00:00",
                "joindate": "2009-04-01 00:00:00",
                "position": "业务员",
                "standard": "1500",
                "depid": "1"
            },
            {
                "empid": "2",
                "username": "小王",
                "sex": "1",
                "mobile": "123456789",
                "birthday": "1979-06-08 00:00:00",
                "joindate": "2009-04-01 00:00:00",
                "position": "技术员",
                "standard": "2500",
                "depid": "2"
            }
        ]
    }
}
```

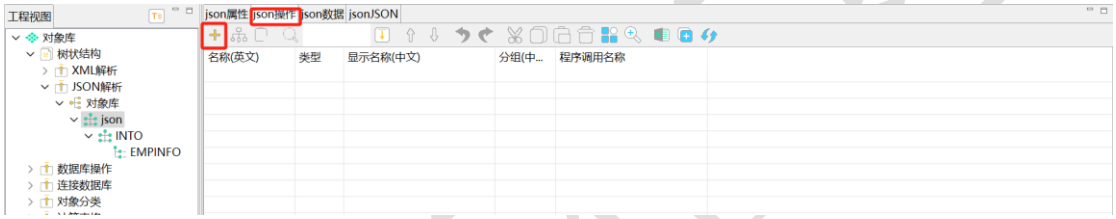
在弹出框中选择 json 文件，点击打开自动生成树结构：



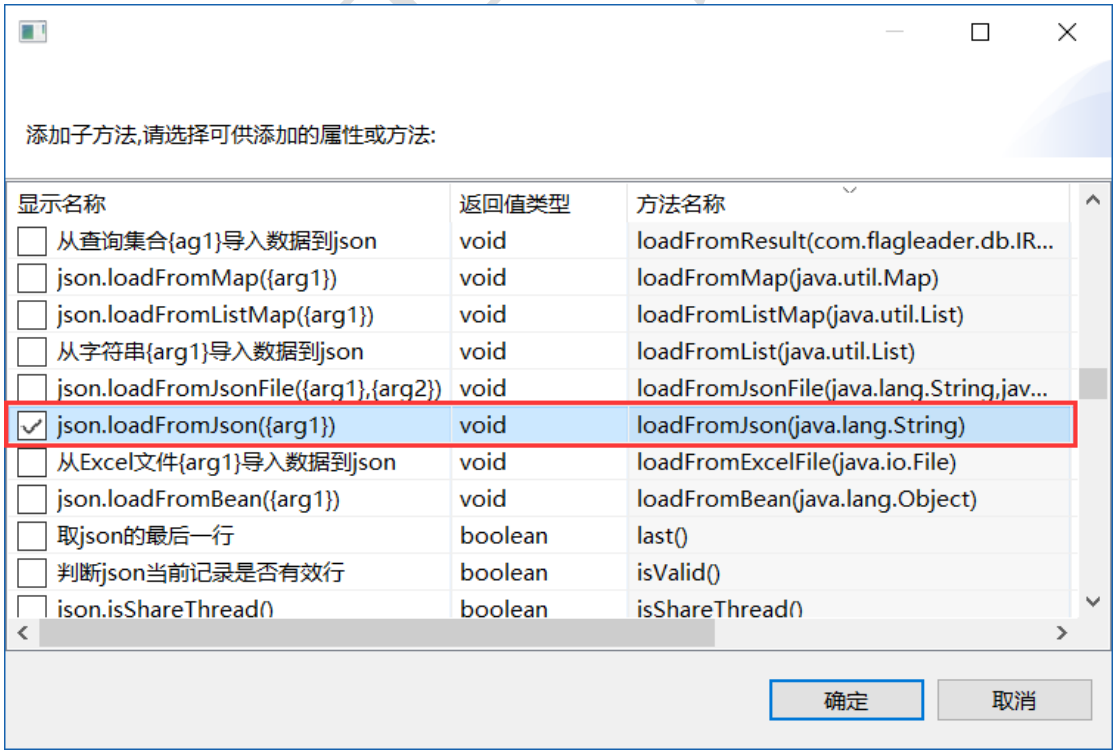
生成的树结构如下图所示：



在树结构“json”的操作中添加方法：

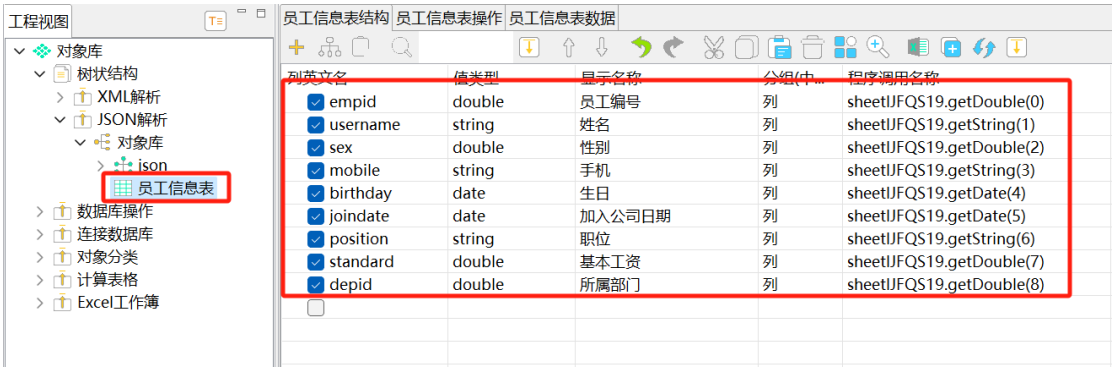


勾选此方法，点击确定添加到操作：

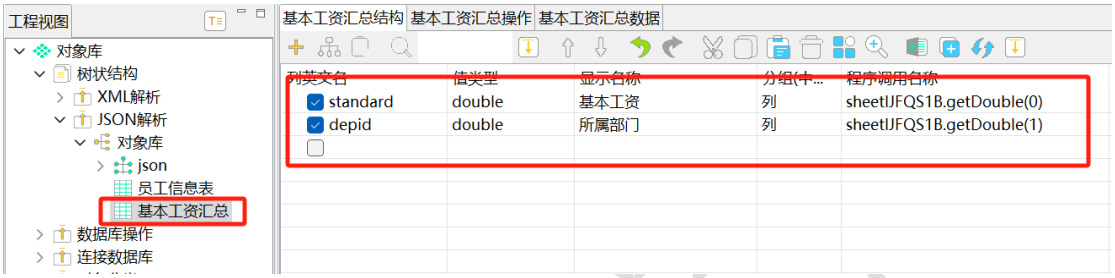


4．添加计算表格

右键对象库，添加第一个计算表格，命名为“员工信息表”，并在表中添加如下字段：



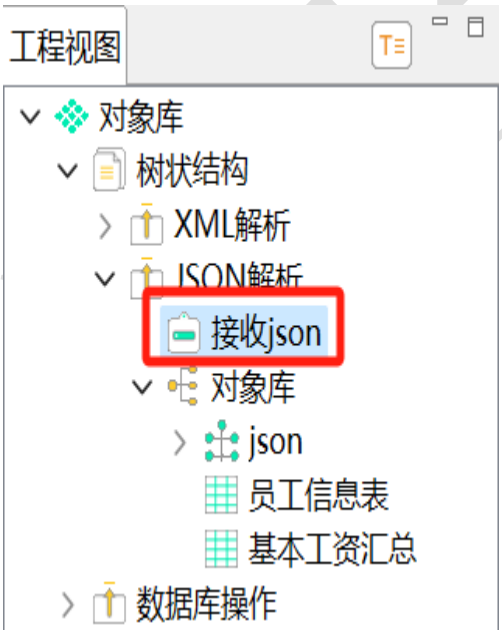
右键对象库，添加第二个计算表格，命名为“基本工资汇总”，并添加表字段：



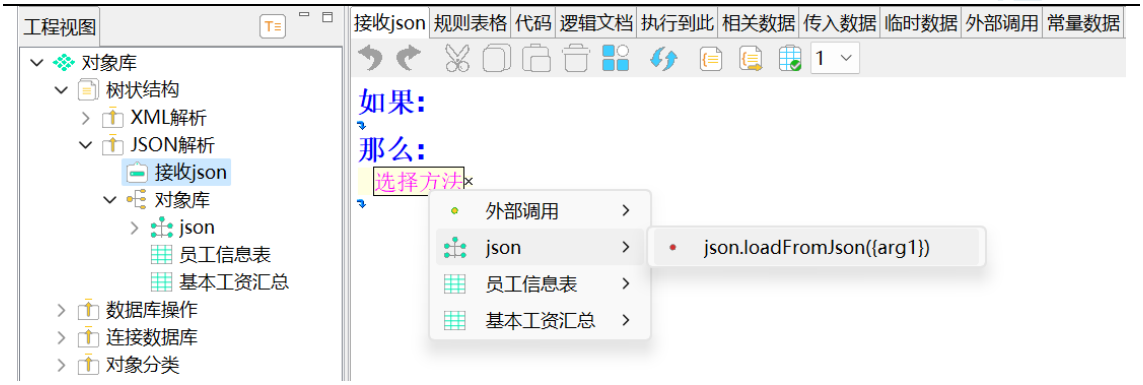
5．新建规则

（1）创建“接收 json”规则

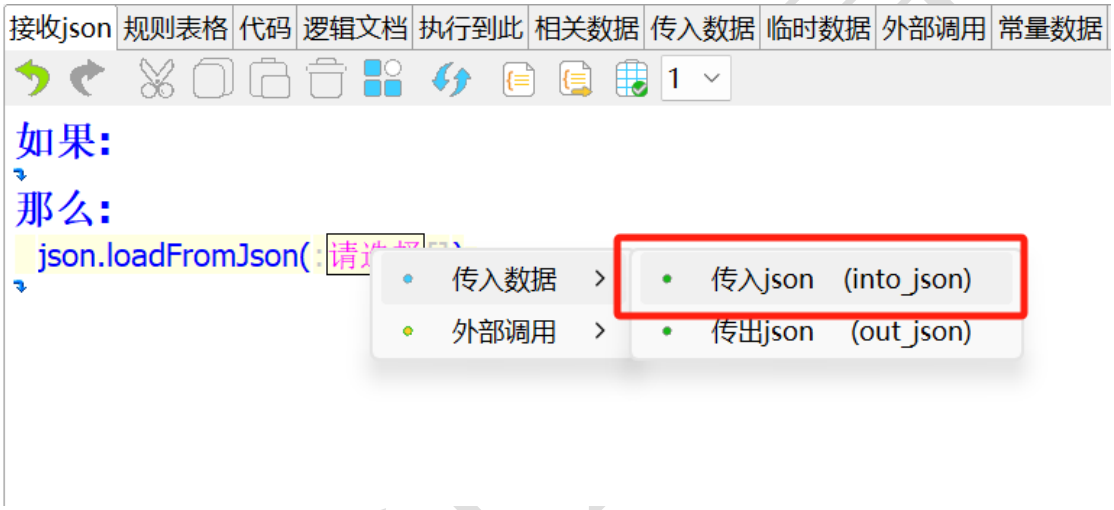
新建规则“接收 json”，用于接收 json 数据，并解析到树结构“json”中，点击“JSON 解析”规则包，添加规则，命名为“接收 json”：



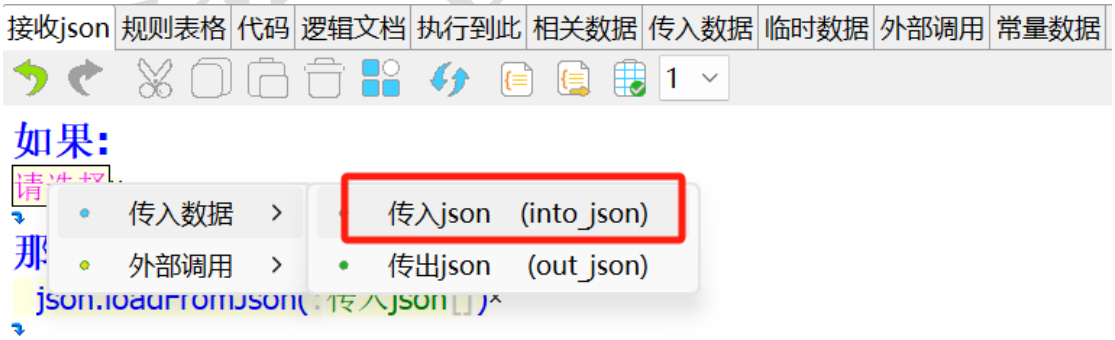
在“接收 json”规则中，点击添加动作，执行方法→json→json.loadFromJson({arg1})：



点击“点击请选择[]”选择 选择值→传入 json

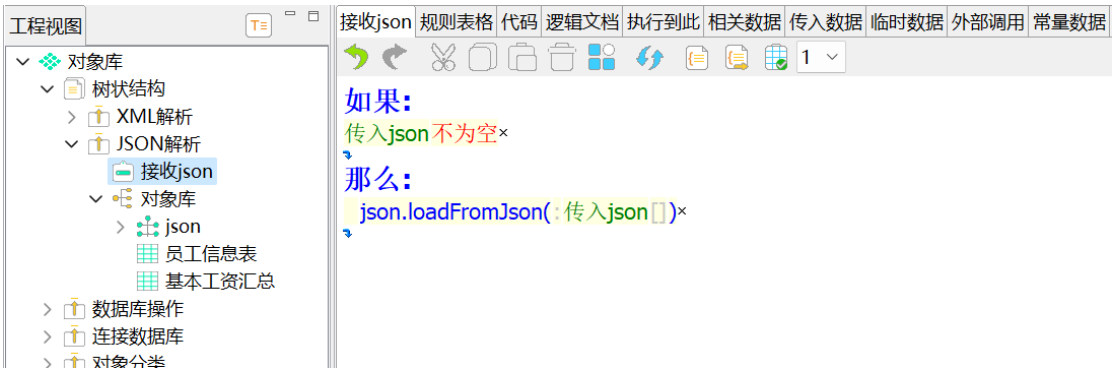


在此之前我们需要对“传入 json”进行判断，如果其值不为空，则才能执行下面的操作。点击“如果”下面的“”添加动作，选择传入 json：



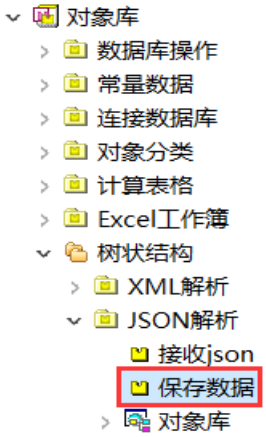
点击“等于”在弹出菜单中选择“不为空”：

“接收 json”规则编写完成后如下图：



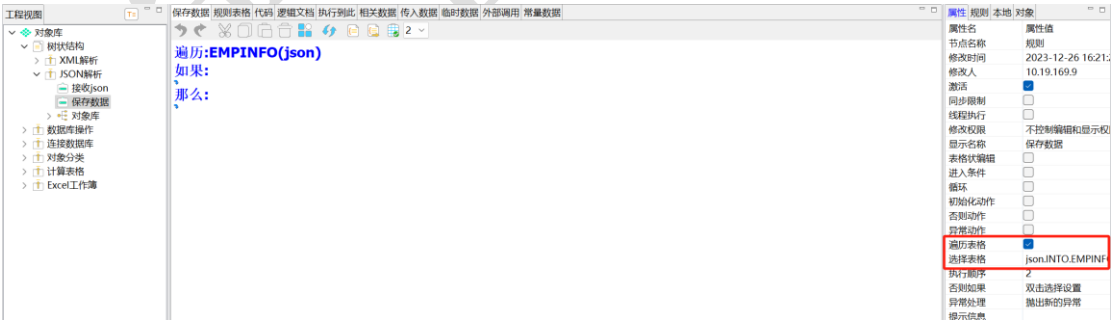
(2) 创建“保存数据”规则

右键“JSON 解析”规则包，添加规则，命名为“保存数据”：

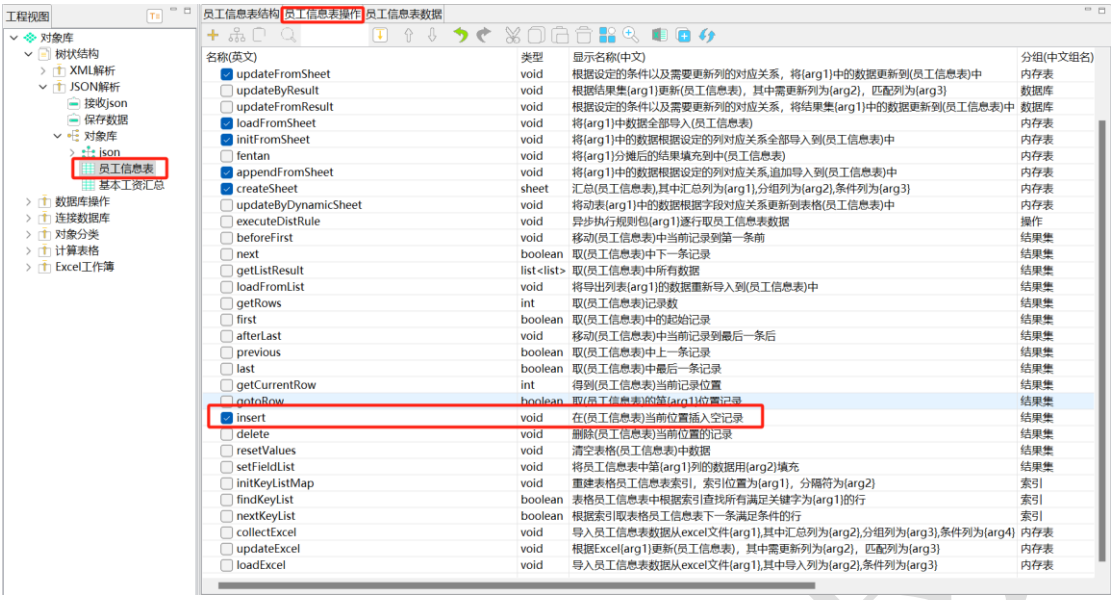


要使“员工信息”中的数据全部保存到“员工信息表”，这需要对“员工信息”进行遍历，从而逐条添加到内存表。

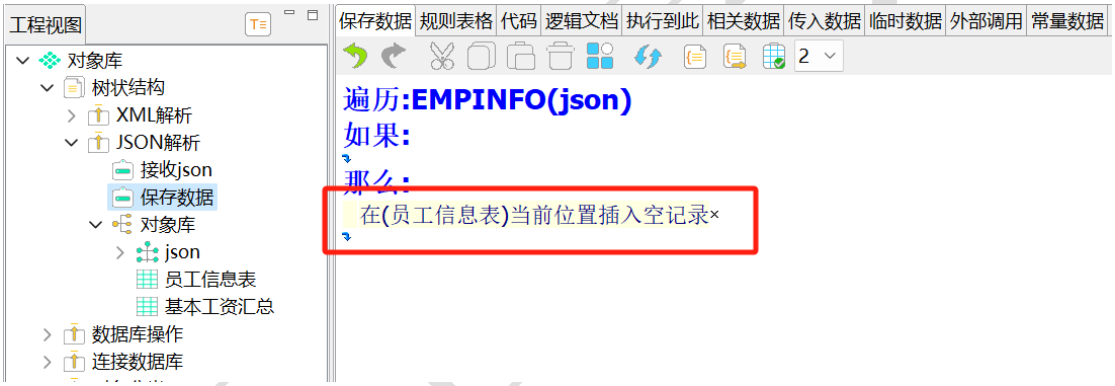
点击“保存数据”规则，在右侧属性窗口将“遍历表格”勾选，下方出现一行“选择表格”，这里我们需要选择“json.INFO.EMPINFO”：



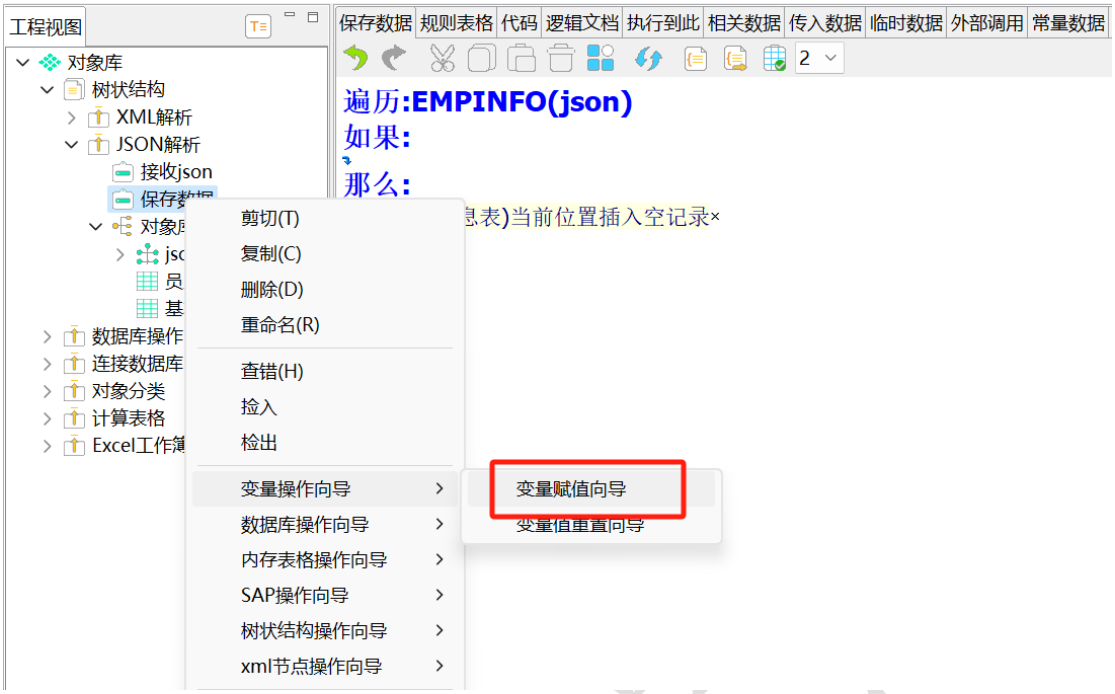
要将一个表中的数据保存到另一张表中，首先需要有插入动作。在“员工信息表”的“员工信息表操作”中，选择 insert 插入方法复制：



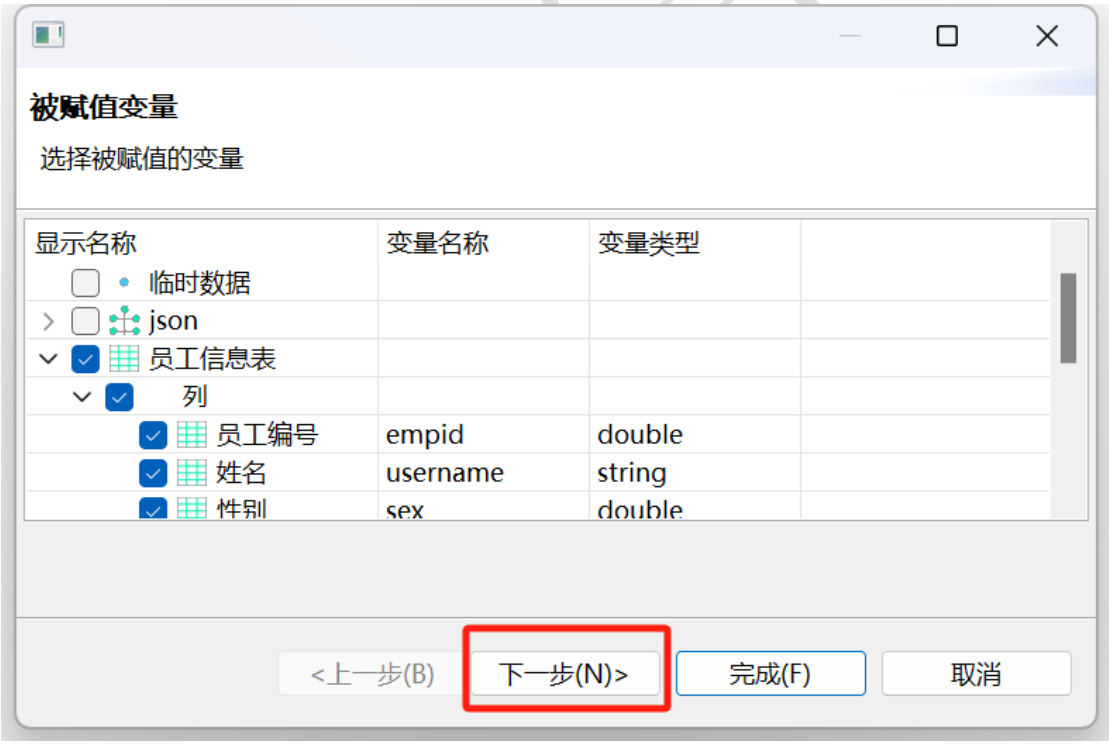
在“保存数据”规则的编辑窗口中，右键添加动作，选择粘贴：



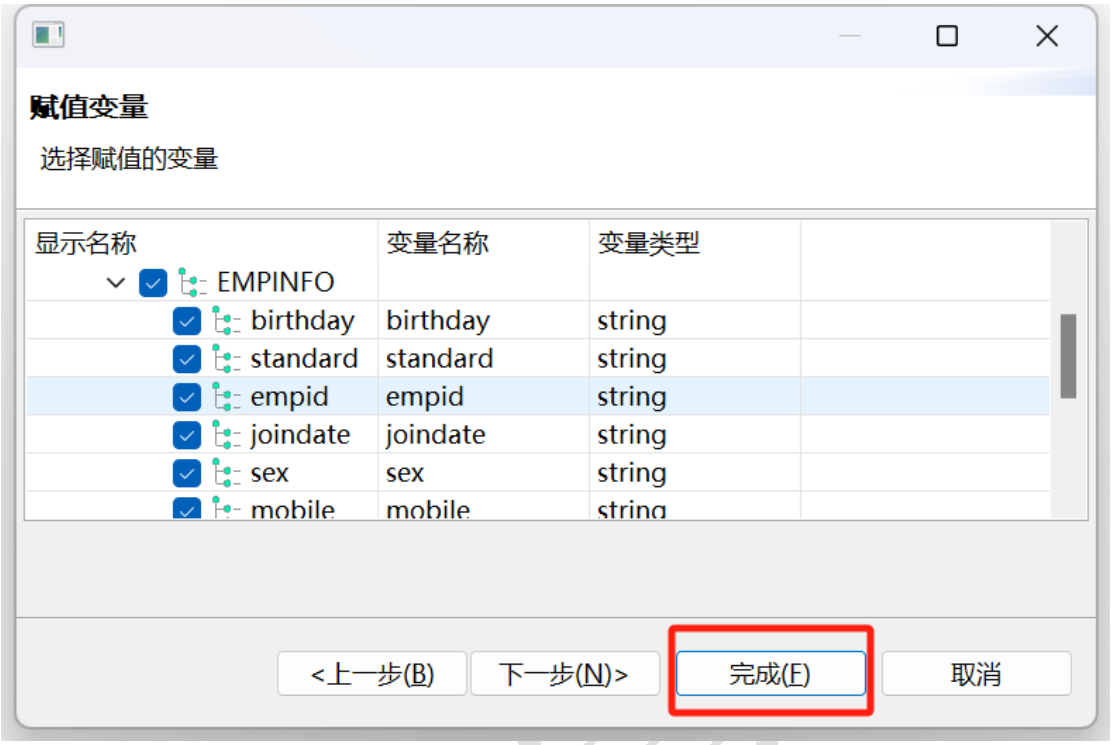
右键“保存数据”规则，选择“变量操作向导”下的“变量赋值向导”：



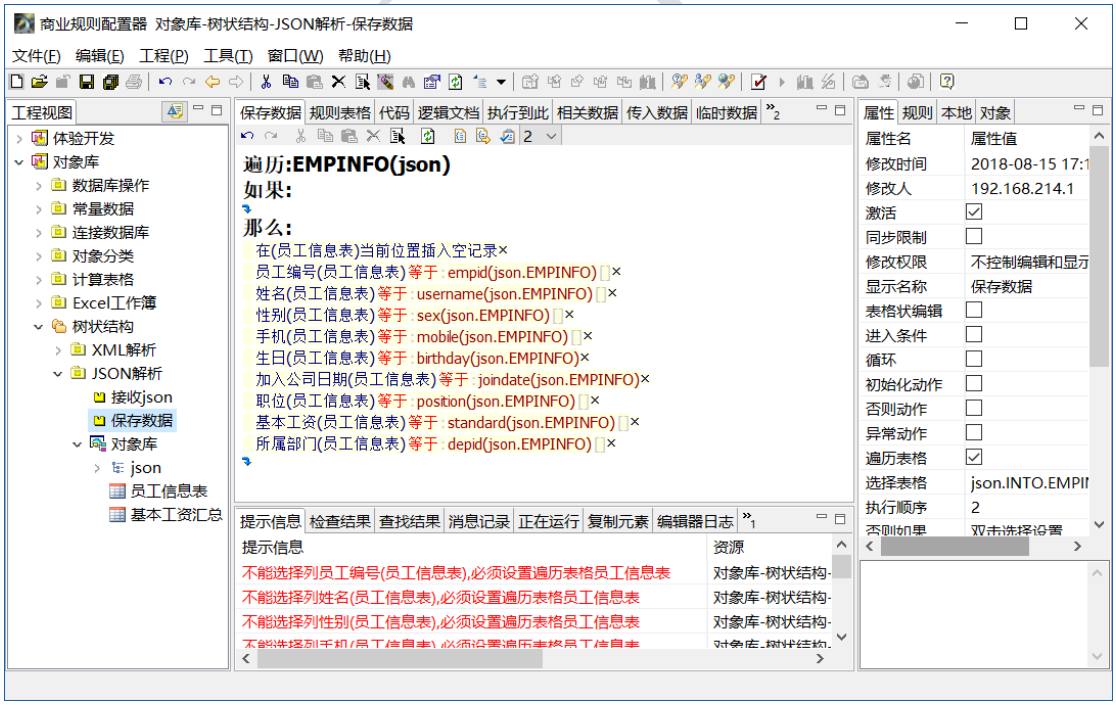
勾选“员工信息表”，点击下一步：



勾选“json”下的“EMPINFO”，点击完成：



点击完成之后，自动生成给表字段赋值操作，这时在下方消息窗口中会出现红色字体报错，提示没有遍历员工信息表：

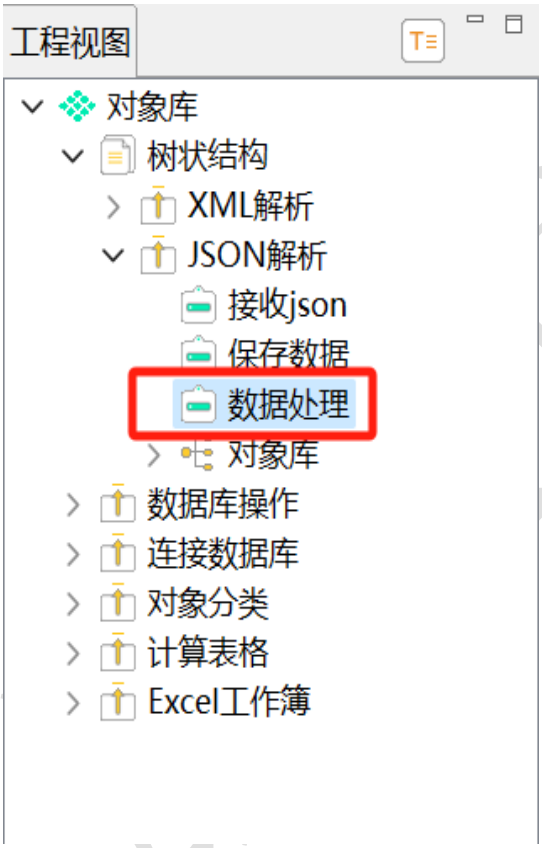


在“保存数据”规则的右侧属性编辑窗口中将“其他表格”的属性值由“不允许设置其他表格的列”更改为“允许设置其他表格的列”：

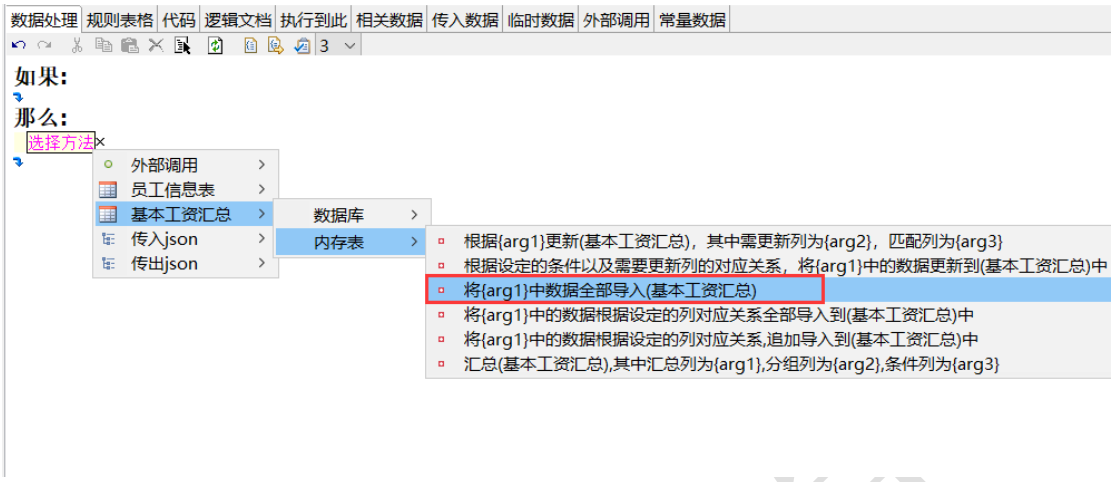
更改好后，点击编辑窗口工具栏中的刷新按钮，下方消息窗口的错误提示消失：

(3) 创建“数据处理”规则

点击“JSON 解析”规则包，右键添加规则，并命名为“数据处理”：

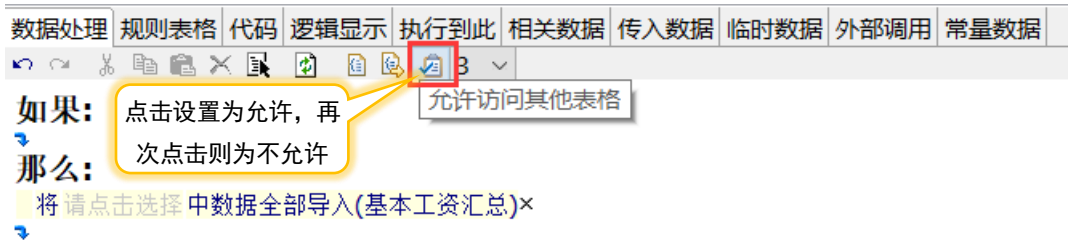


点击添加动作，执行方法→选择方法→基本工资汇总→内存表→将{arg}中数据全部导入（基本工资汇总）

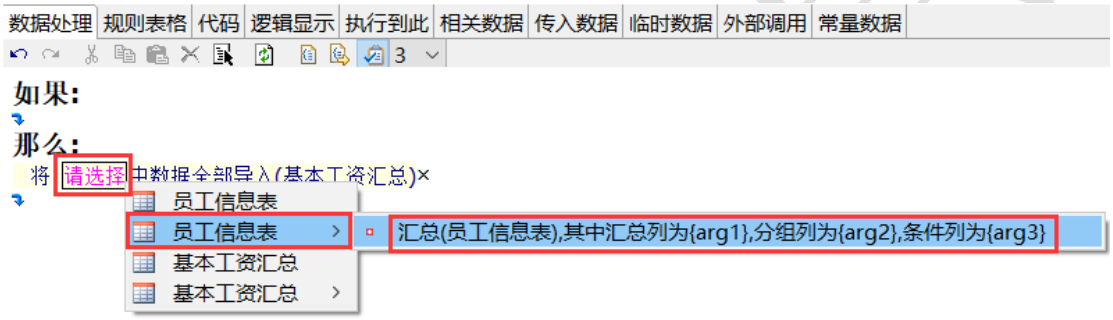


在属性编辑窗口中，将“其他表格”的属性值改为“允许设置其他表格的列”：

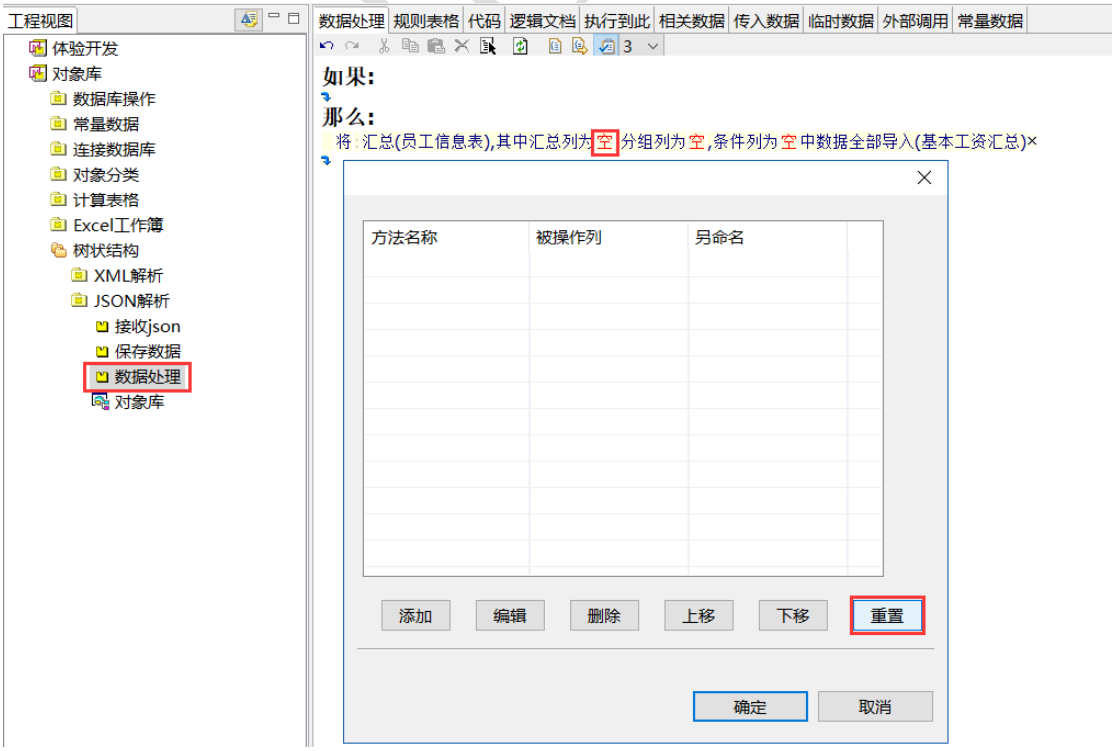
或者在编辑窗口，点击工具栏上的“ ”按钮，设置为“允许设置其他表格的列”：

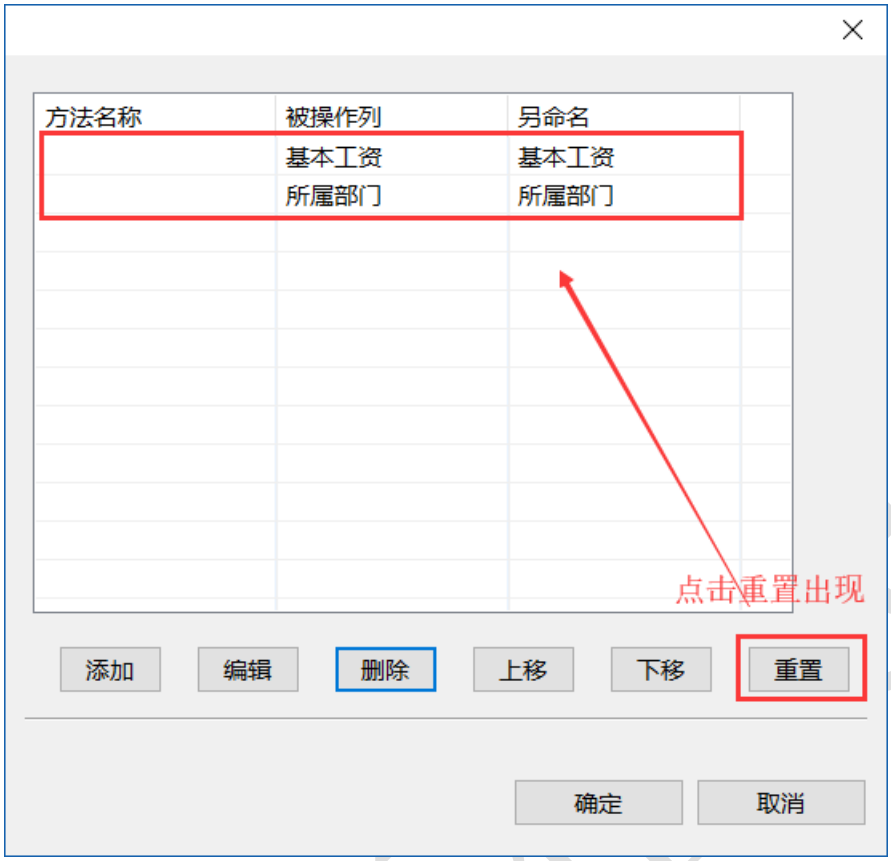


点击“点击请选择[]”选择“选择值”→员工信息表下的方法，如图：

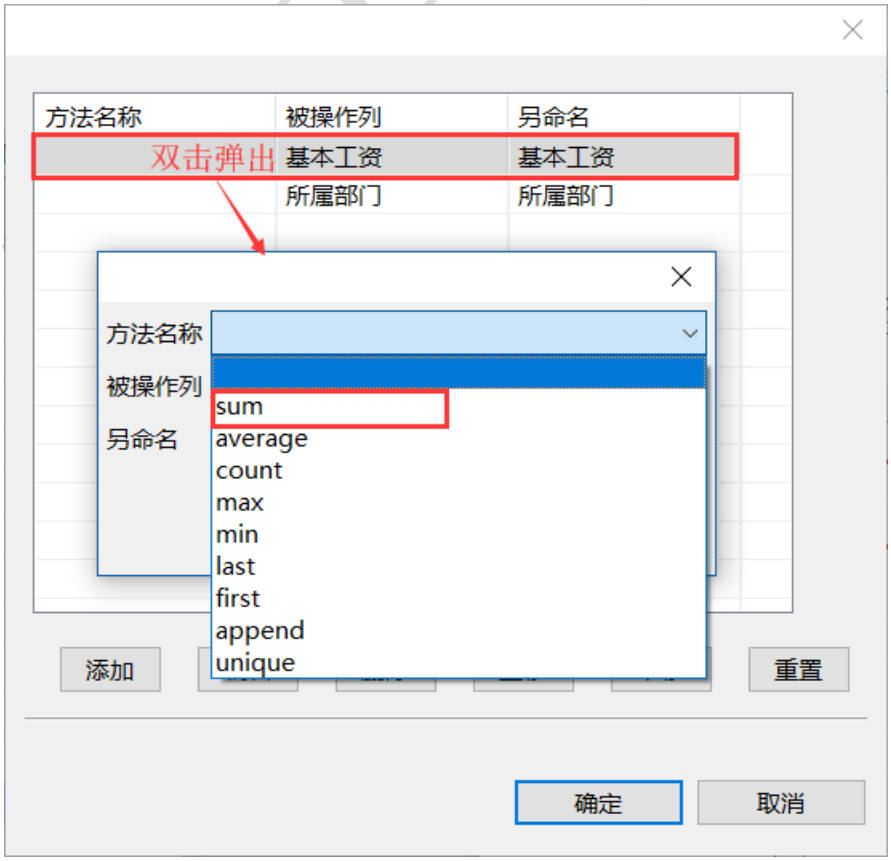


点击第一个空，选择汇总的列，在弹出框中点击重置按钮，自动添加所匹配到的列：

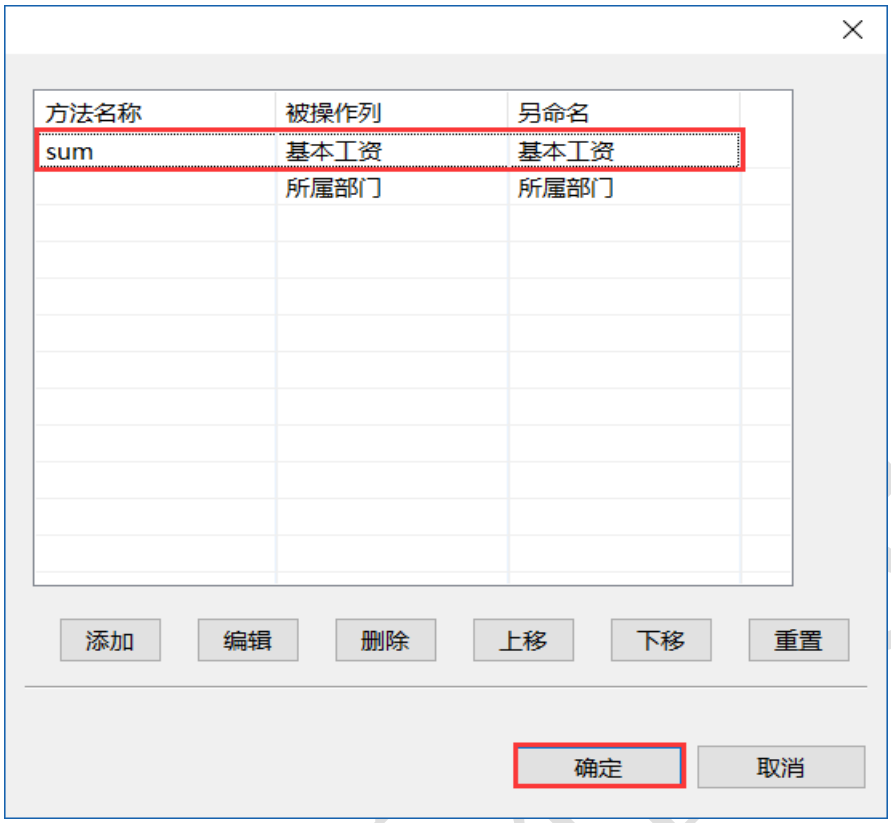




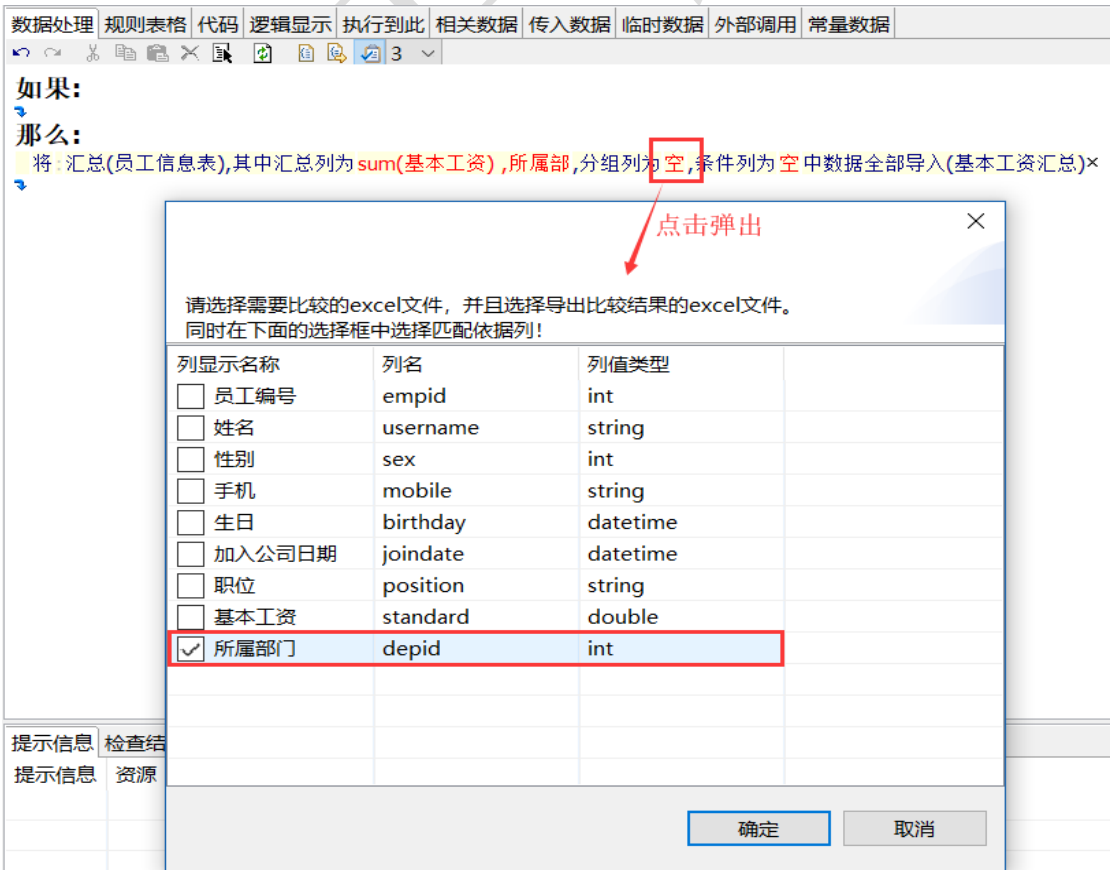
双击“基本工资”这一行，在弹出框中选择“方法名称”点击下拉，选择“sum”方法：



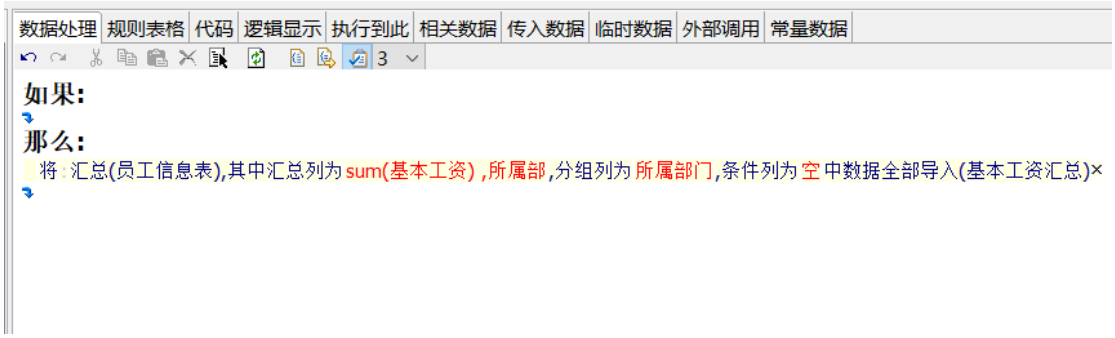
完成后点击确定，基本工资方法添加完成：



点击第二个空，添加分组列，在弹出框中将“所属部门”勾选：

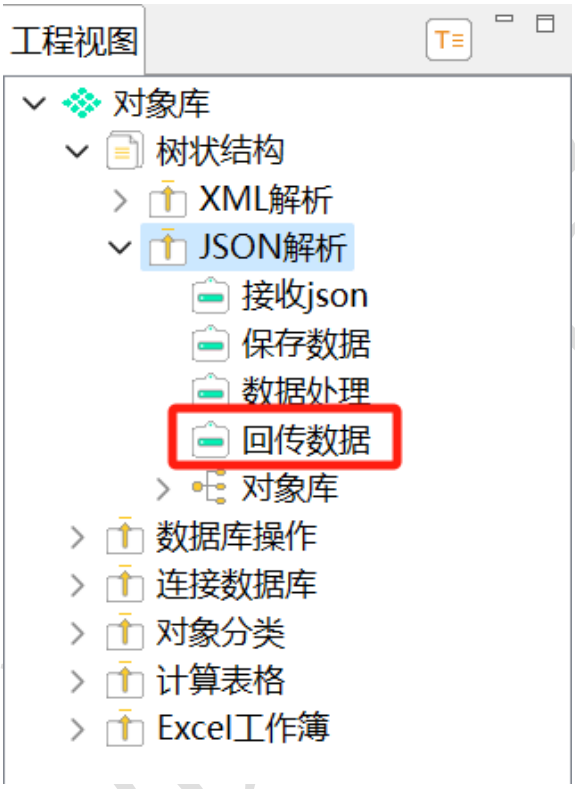


完成后规则如下：

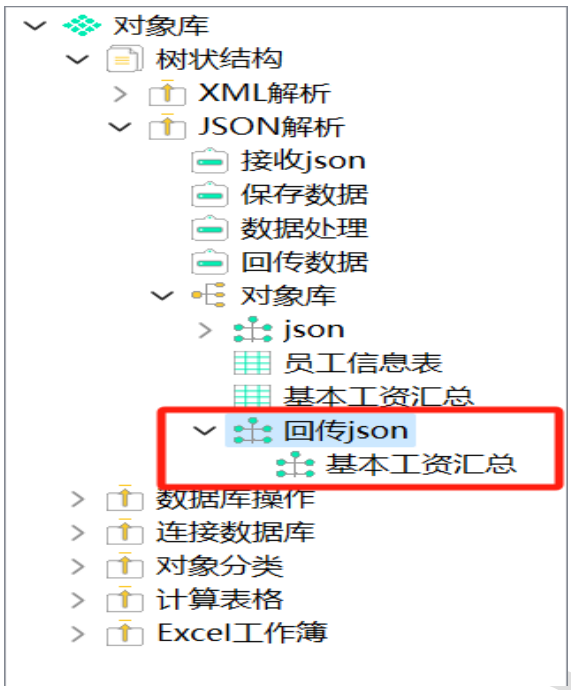


(4) 创建“回传数据”规则

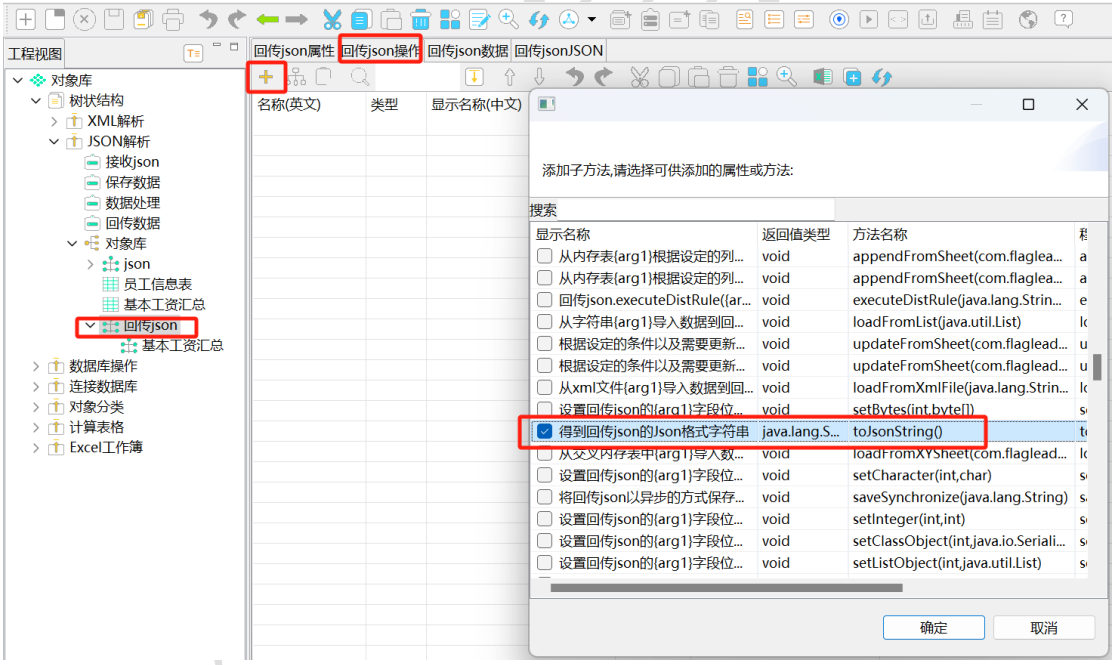
右键“JSON 解析”规则包，添加规则，并命名为“回传数据”：



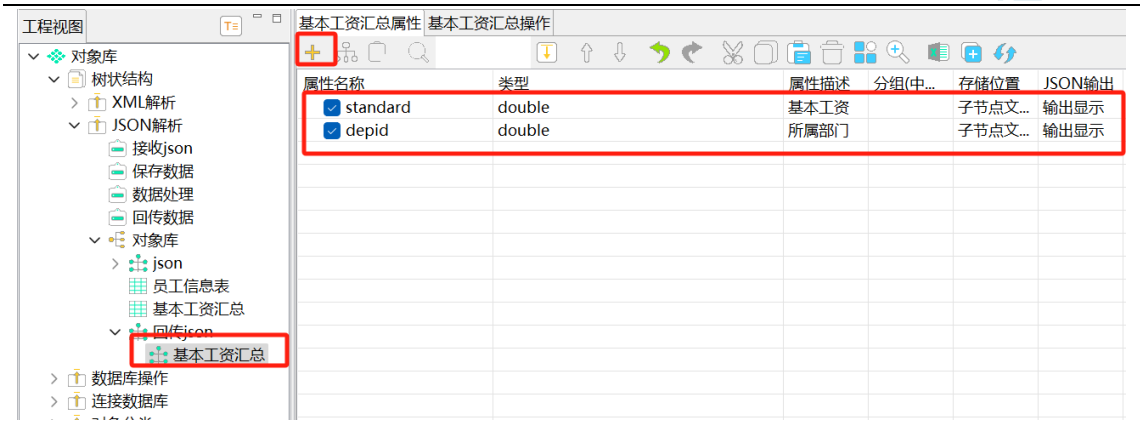
在规则“回传数据”中，我们需要将最终的计算结果值回传到树结构中，因此首先需要添加用于保存回传数据的树结构，添加的树结构如下：



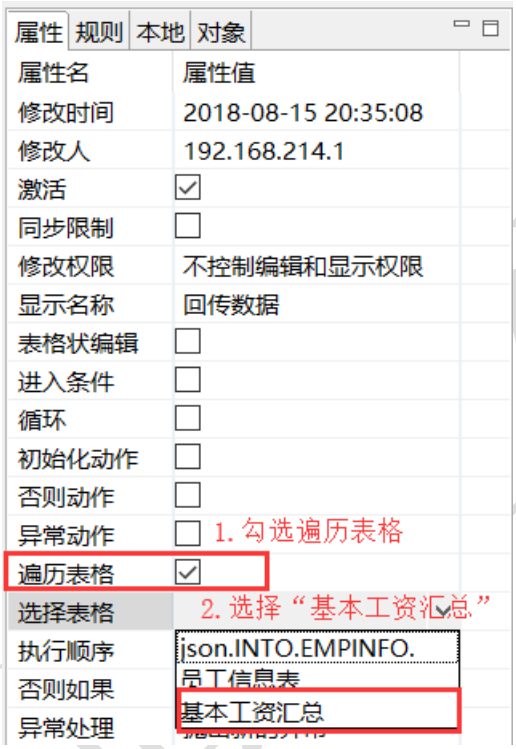
在“回传 json”树结构中需要添加一个方法，添加过程如下所示：



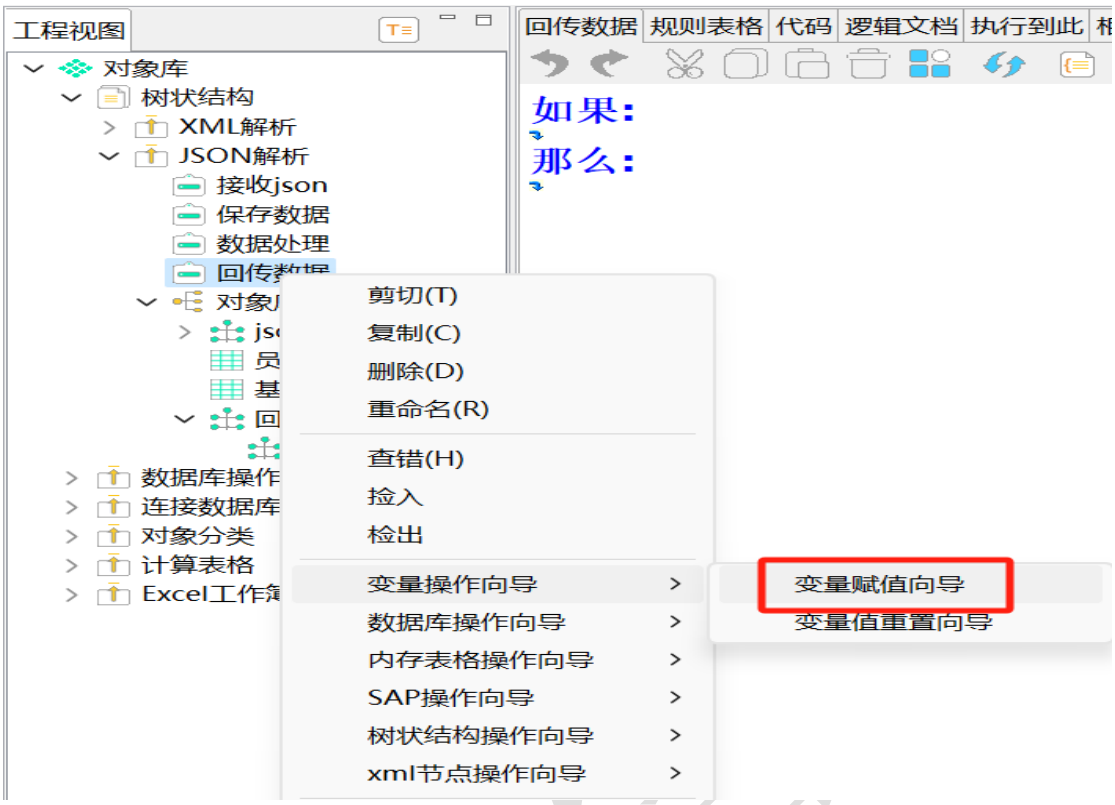
在“基本工资汇总”中添加如下所示变量：



树结构添加完成之后，开始编写规则逻辑。首先在规则“回传数据”的属性窗口中设置遍历表格“基本工资汇总”表：



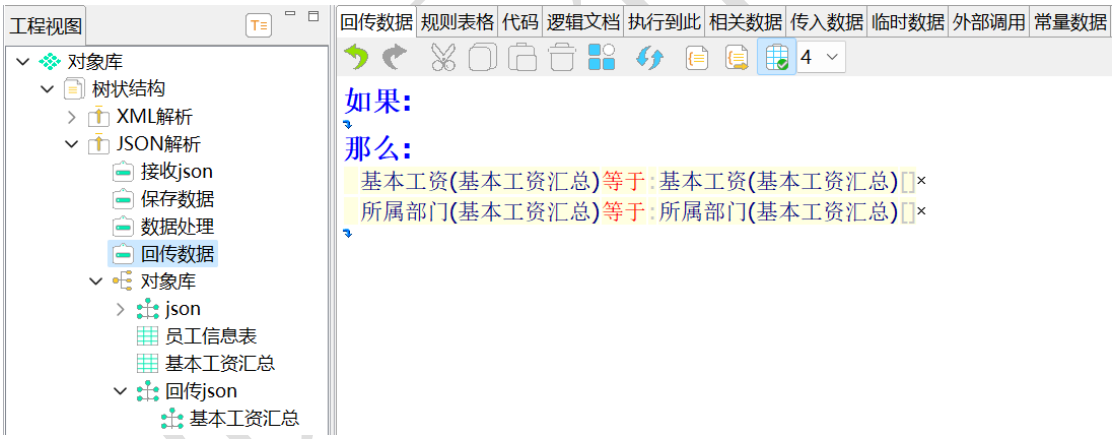
右键“回传数据”规则，选择 变量操作向导→变量赋值向导：



勾选“回传 json”下的“基本工资汇总”，点击下一步：

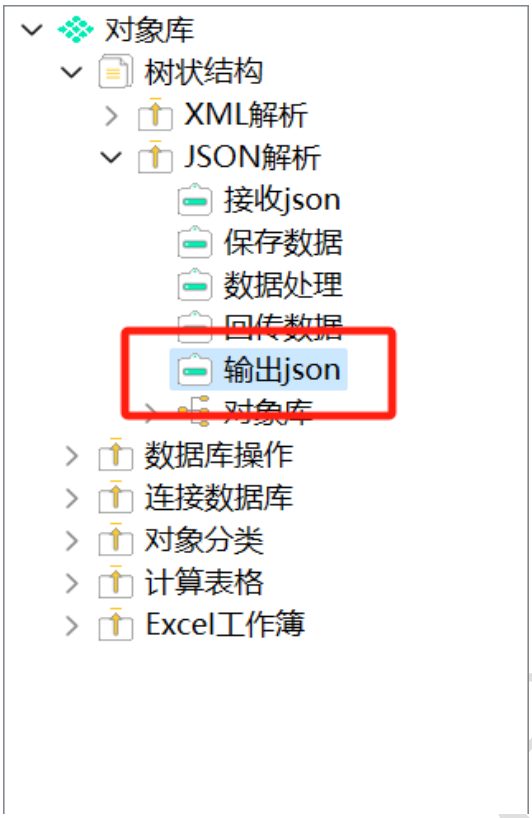
勾选“基本工资汇总”，点击完成。

完成之后如下图：

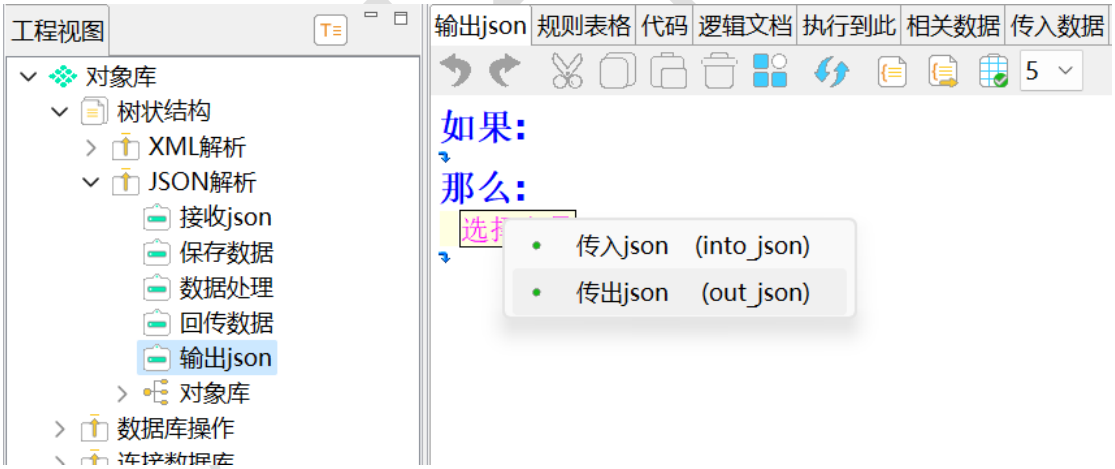


(5) 创建“输出 json”规则

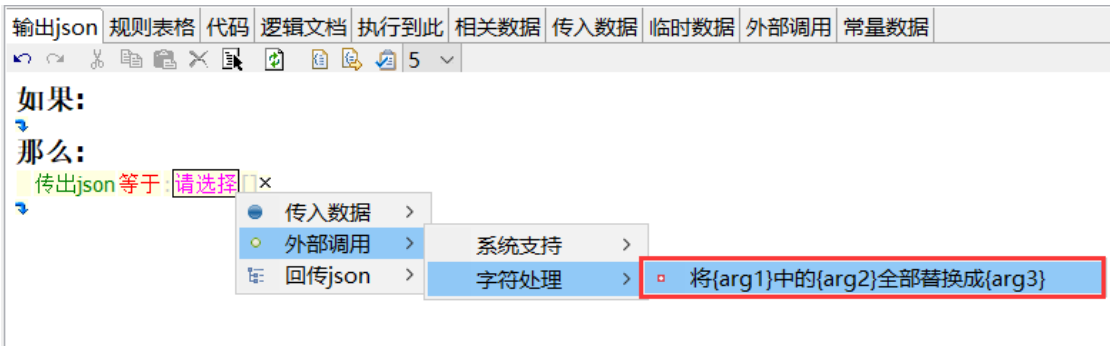
点击“JSON 解析”规则包，右键添加规则，命名为“输出 json”：



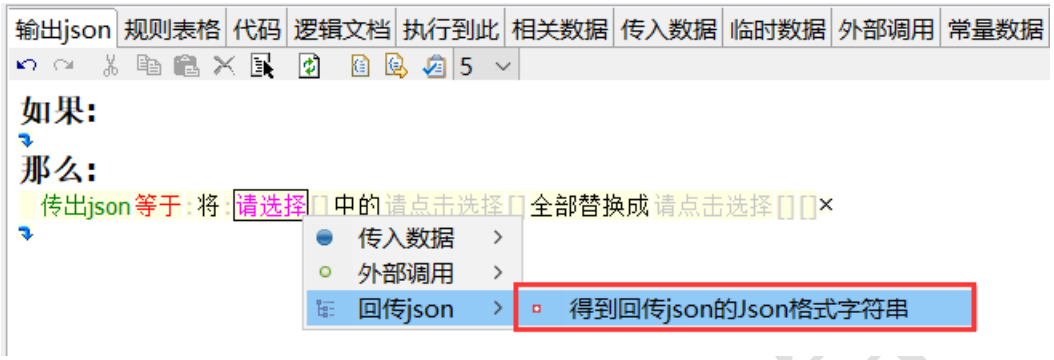
点击添加动作，变量赋值→传出 json：



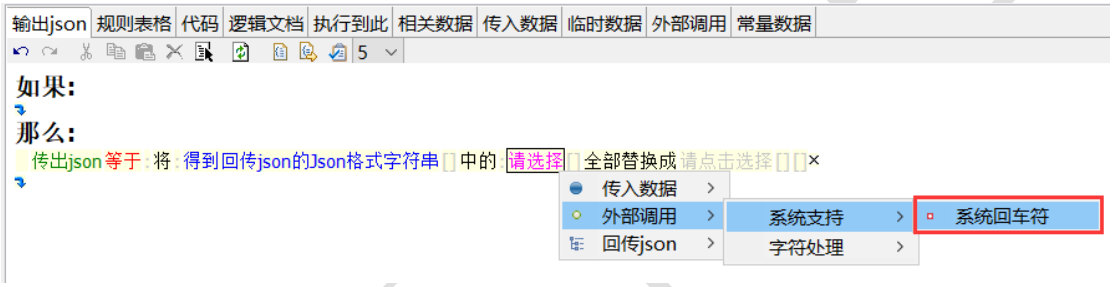
点击“点击请选择[]”，选择值→外部调用→字符处理：



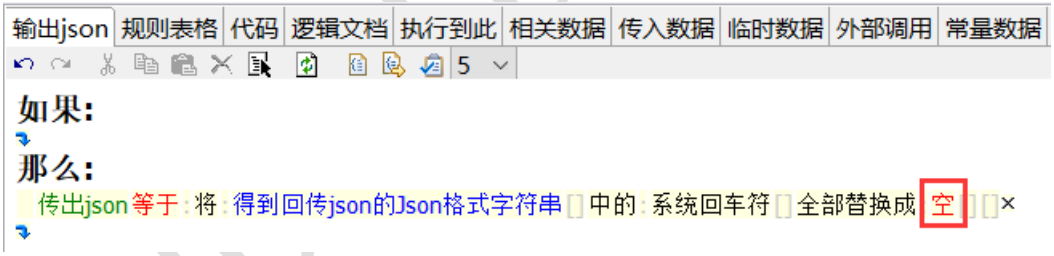
点击第一个“请点击选择[]”，选择 选择值→回传 json：



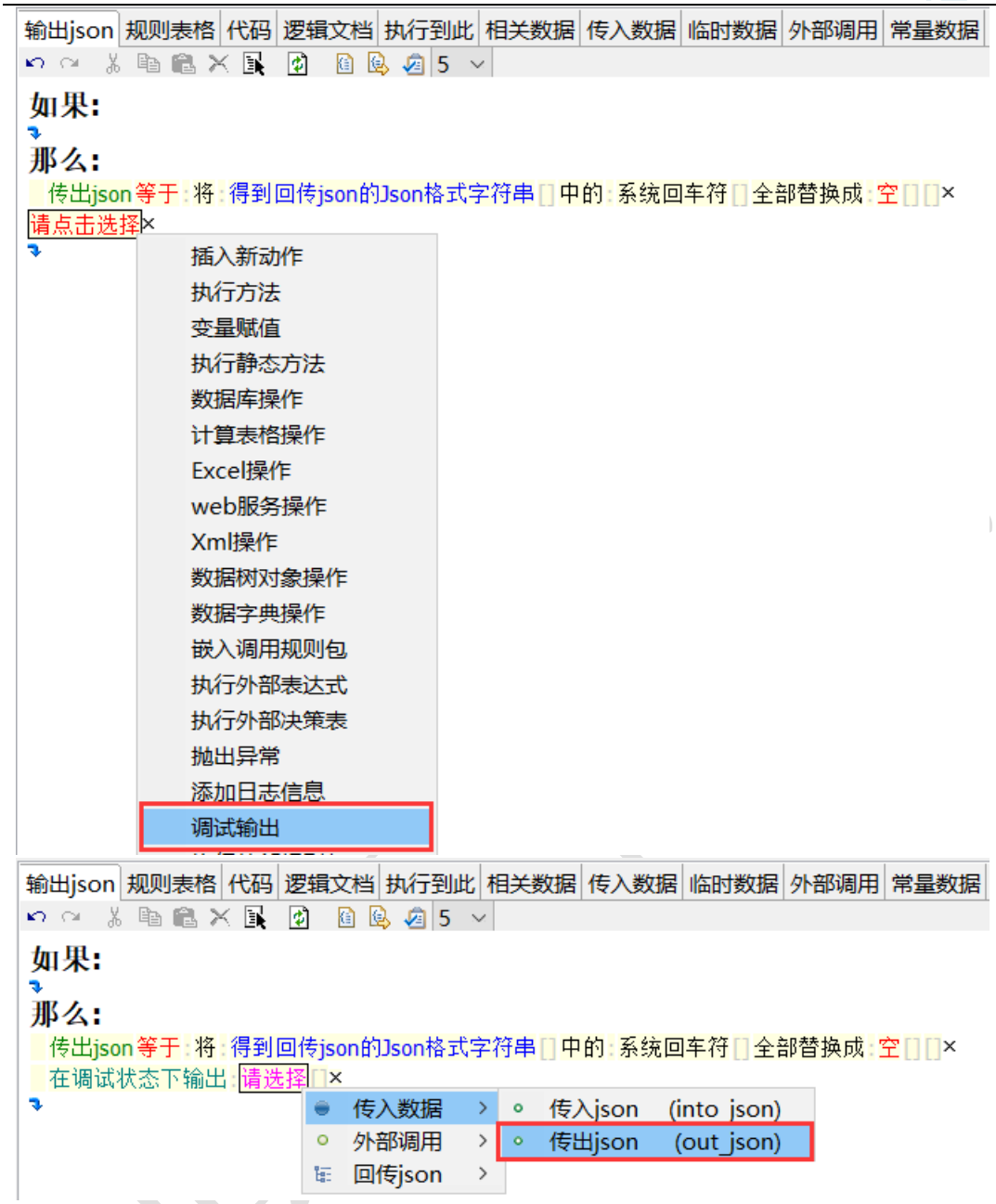
点击第二个“请点击选择[]”，选择 选择值→外部调用→系统支持：



点击第三个“请点击选择[]”，选择输入常量值，值为空：



点击添加动作，将“传出 json”输出在控制台消息窗口，选择“调试输出”（等同于 System.out.println()）。



规则编写完成后，点击“”保存并编译。

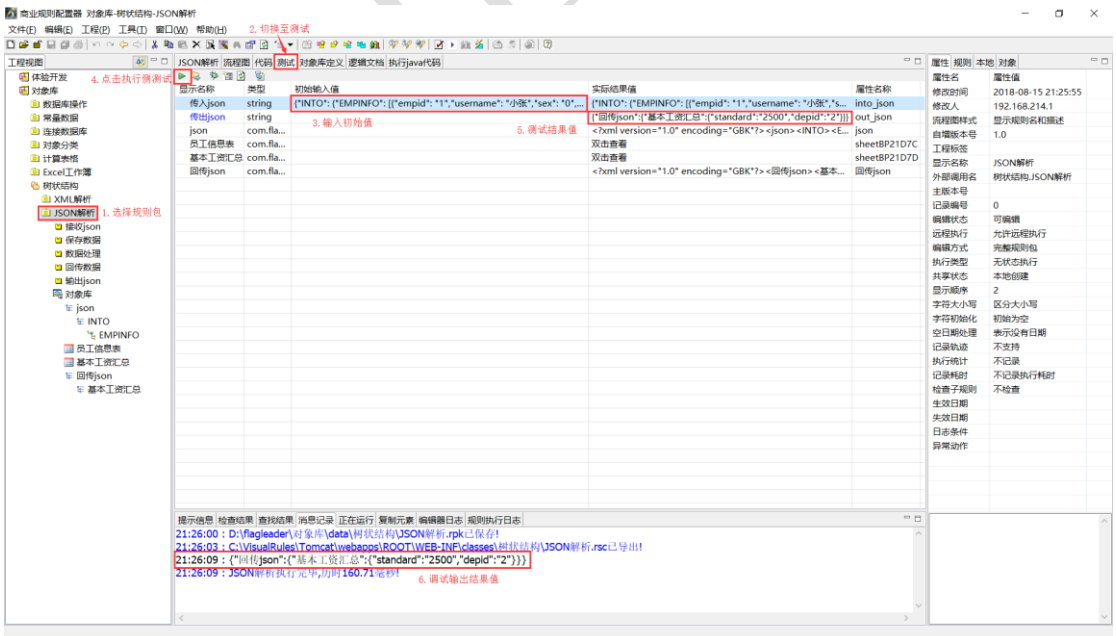
6. 测试

点击“JSON 解析”规则包，在编辑窗口中输入初始值进行测试。

这里的初始值是 json 字符串，因此要测试首先需要编写 json 格式的字符串，如下所示：

```
{
    "INTO": {
        "EMPINFO": [
            {
                "empid": "1",
                "username": "小张",
                "sex": "0",
                "mobile": "123456789",
                "birthday": "1979-06-08 00:00:00",
                "joindate": "2009-04-01 00:00:00",
                "position": "业务员",
                "standard": "1500",
                "depid": "1"
            },
            {
                "empid": "2",
                "username": "小王",
                "sex": "1",
                "mobile": "123456789",
                "birthday": "1979-06-08 00:00:00",
                "joindate": "2009-04-01 00:00:00",
                "position": "技术员",
                "standard": "2500",
                "depid": "2"
            }
        ]
    }
}
```

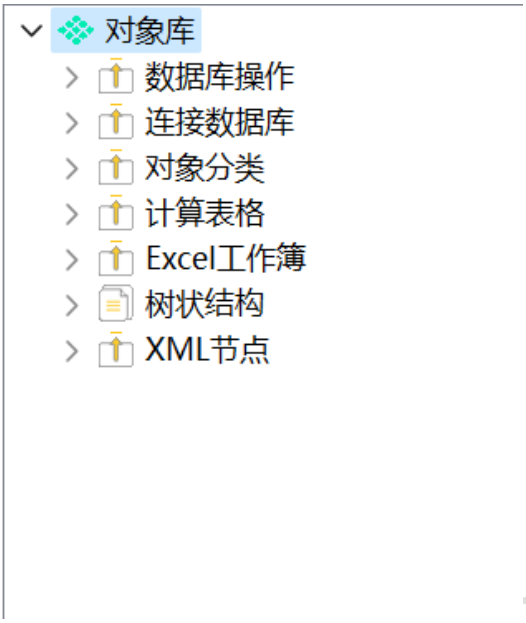
编写完成后需要对其进行格式化，去掉所有的空格和换行，使之变成一行字符串，然后将这行字符串复制到测试中的“初始输入值”一栏中，点击开始测试：



2.3.6. XML 节点

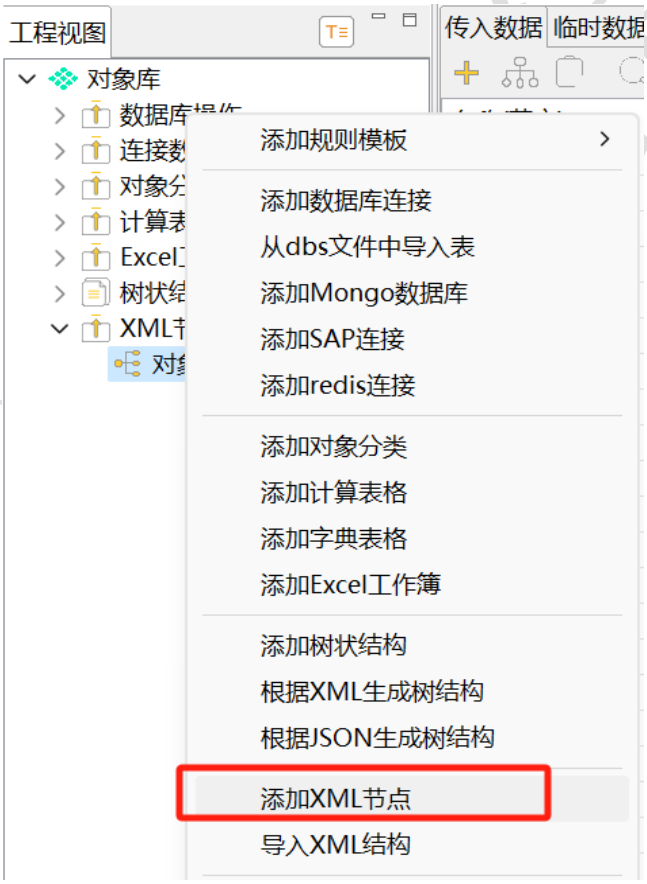
1．创建规则包

点击“对象库”工程，新建规则包，命名为“xml 节点”：

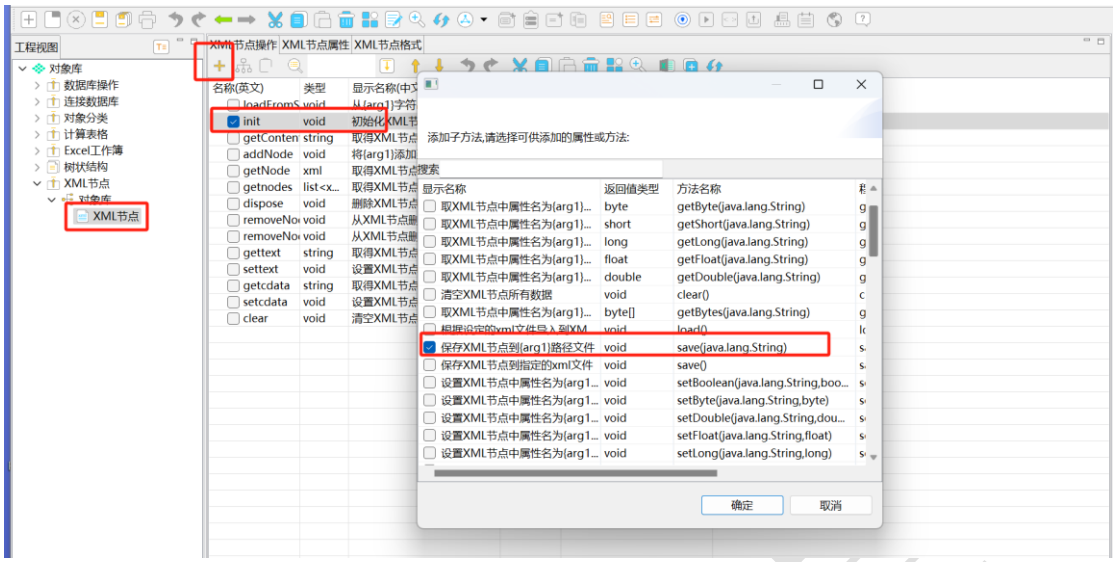


2．添加 xml 节点

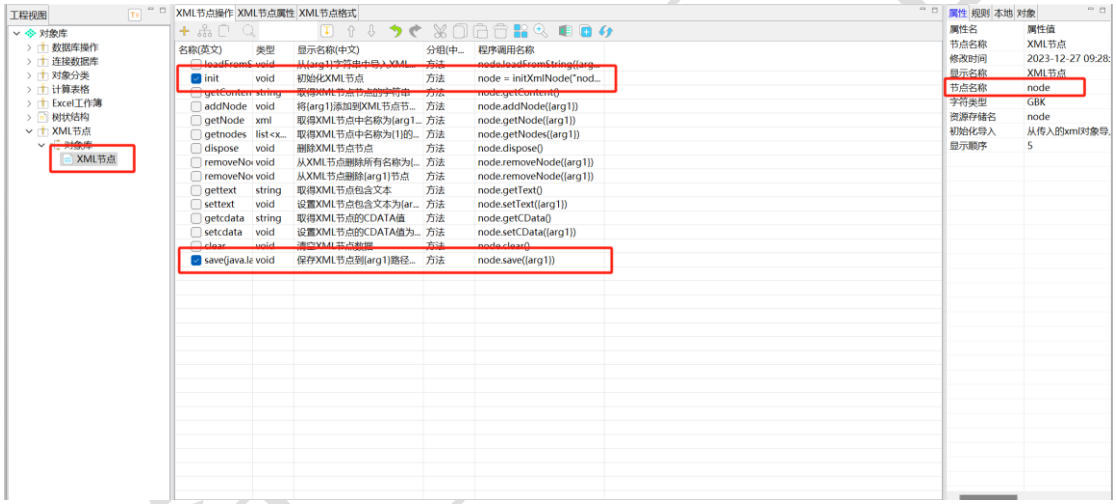
打开“xml 节点”规则包，在对象库下右键“添加 xml 节点”：



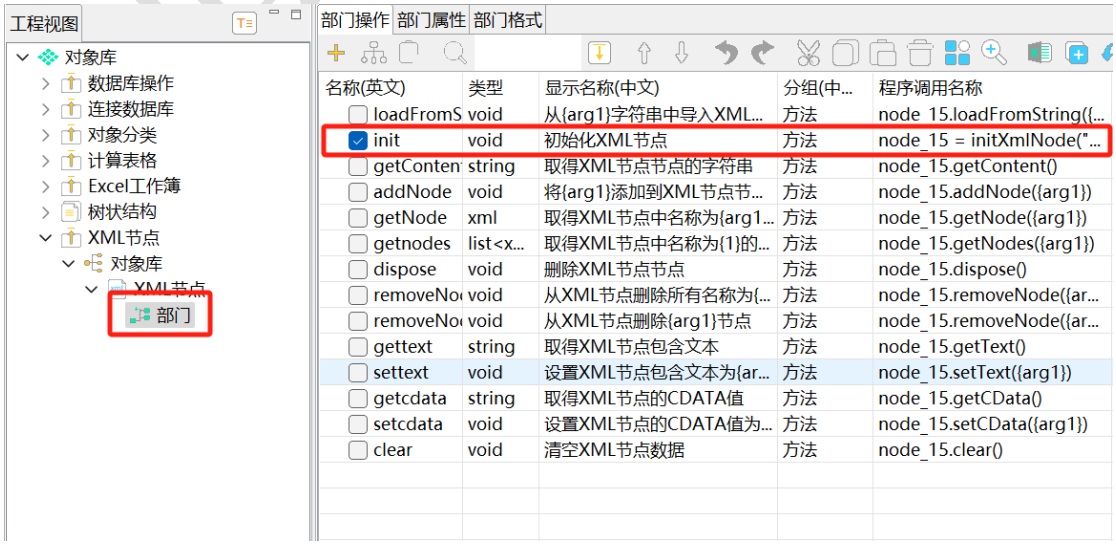
打开刚刚所添加的 xml 节点，在“xml 节点操作”中勾选 init 方法，并在工具栏中点击添加，在弹出框中勾选 save(java.lang.String)方法：



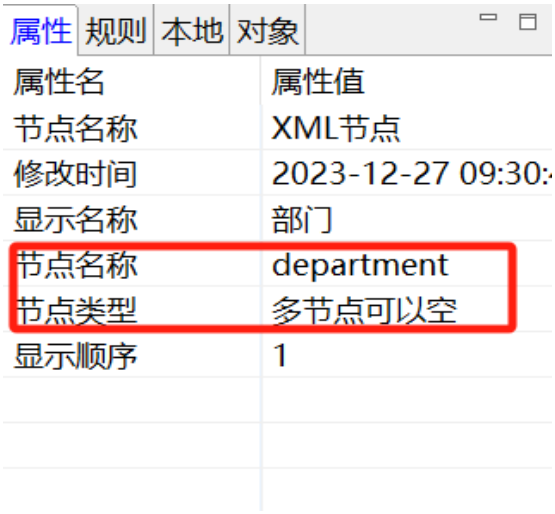
在右侧属性窗口中将“节点名称”的属性值改为英文“node”



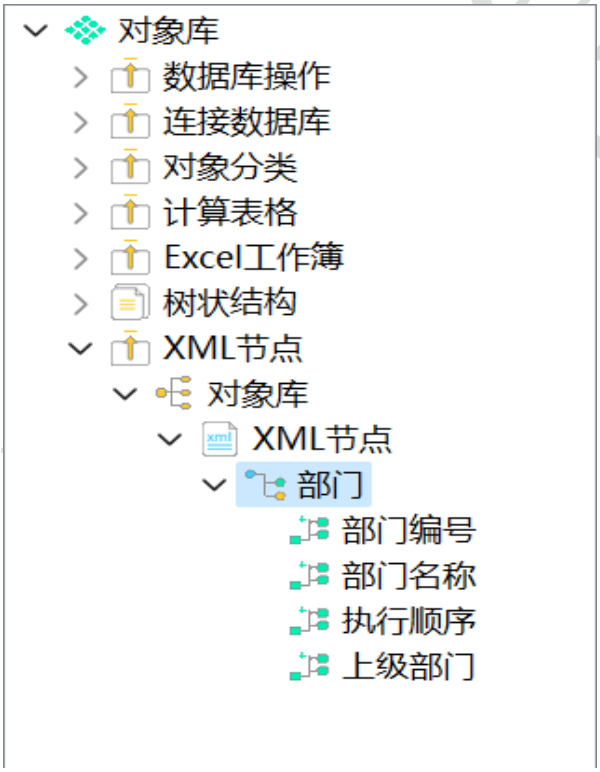
点击“xml 节点”，在此节点下右键“添加 xml 节点”，命名为“部门”，在编辑窗口的“部门操作”中勾选 init 方法：



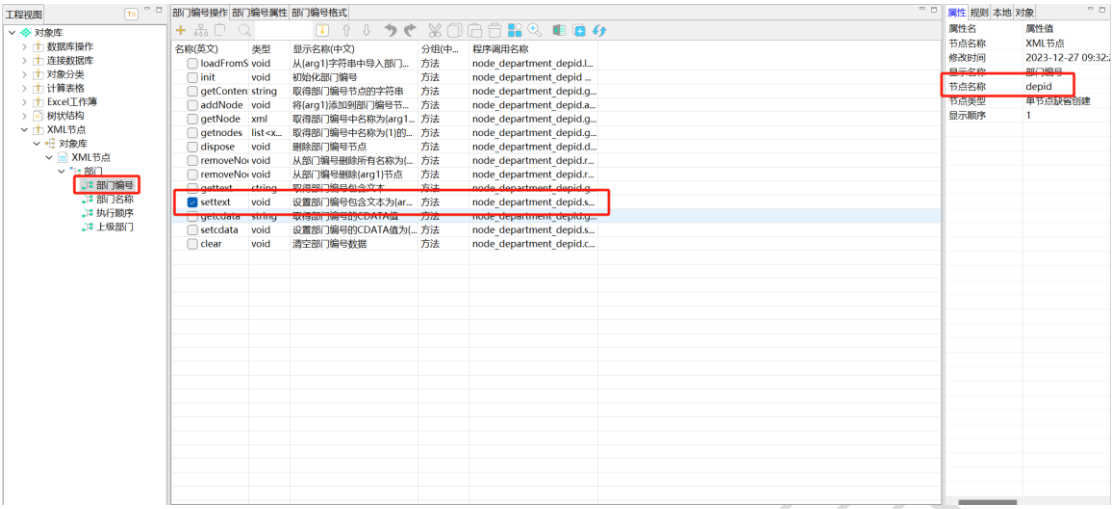
在右侧属性窗口中更改节点名称为“department”，并点击节点类型后的下拉列表，将其值由“单节点缺省创建”改为“多节点可以为空”：



再在“部门”xml 节点下分别添加“部门编号”、“部门名称”、“执行顺序”、“上级部门”xml 节点，如下图：

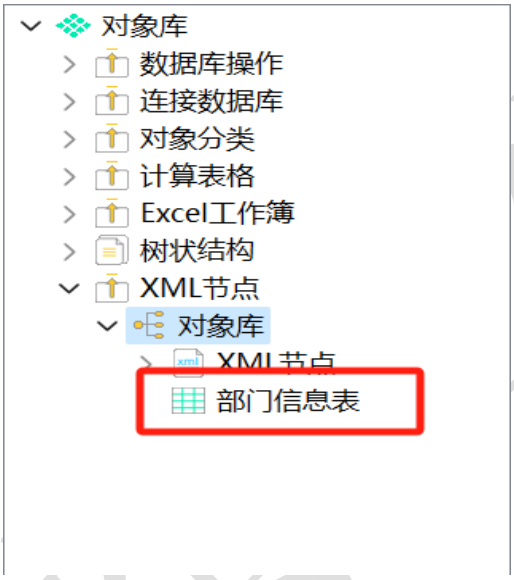


分别在这四个 xml 节点中勾选 settext 方法，并将节点名称依次改为英文名称“depid”、“depname”、“ExecutionOrder”、“SuperDep”：

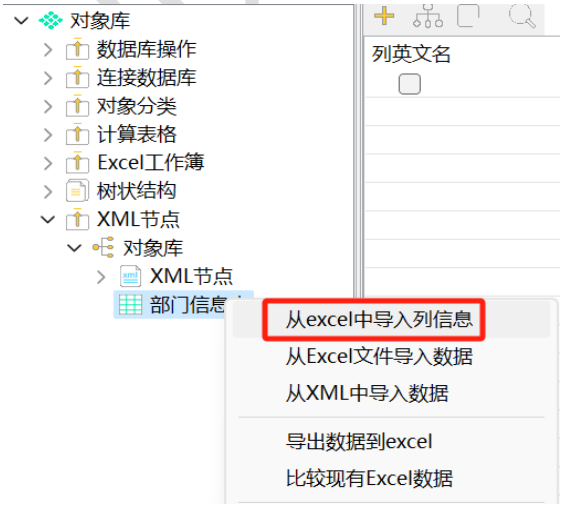


3．添加计算表格

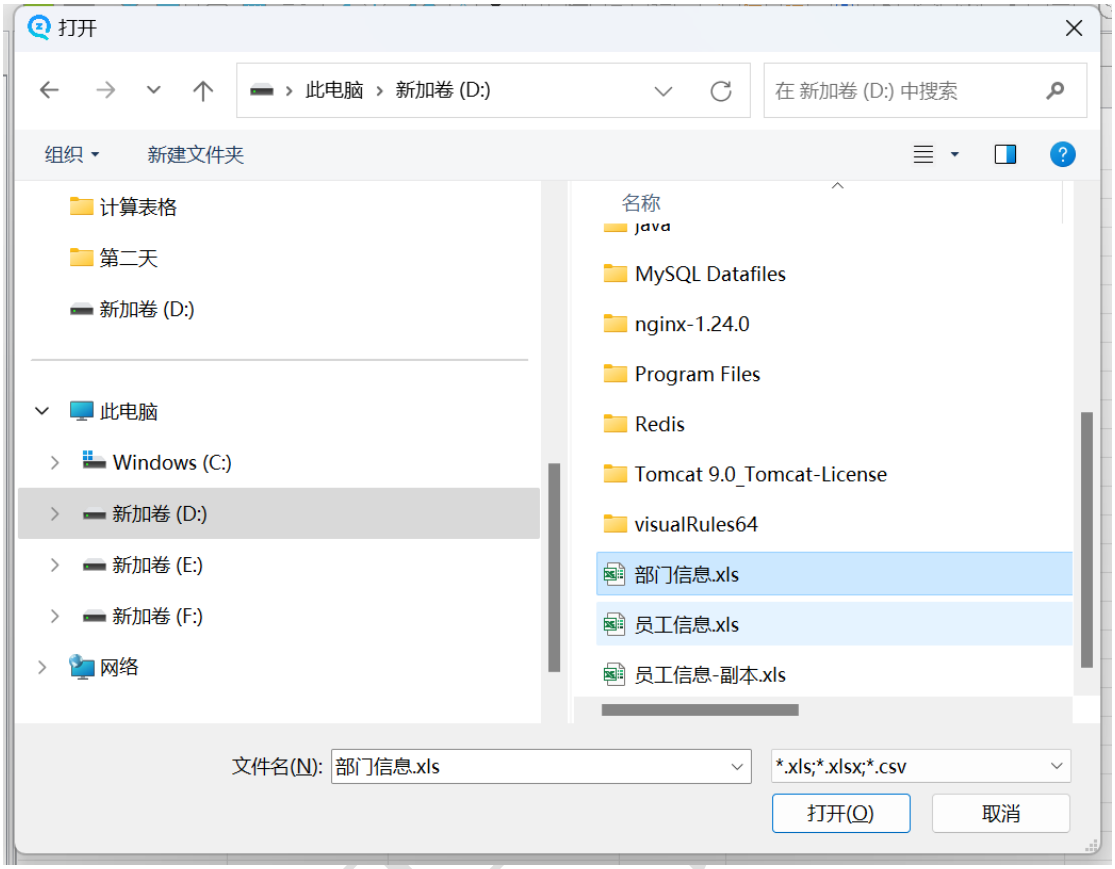
点击对象库，右键添加计算表格，命名为“部门信息表”：



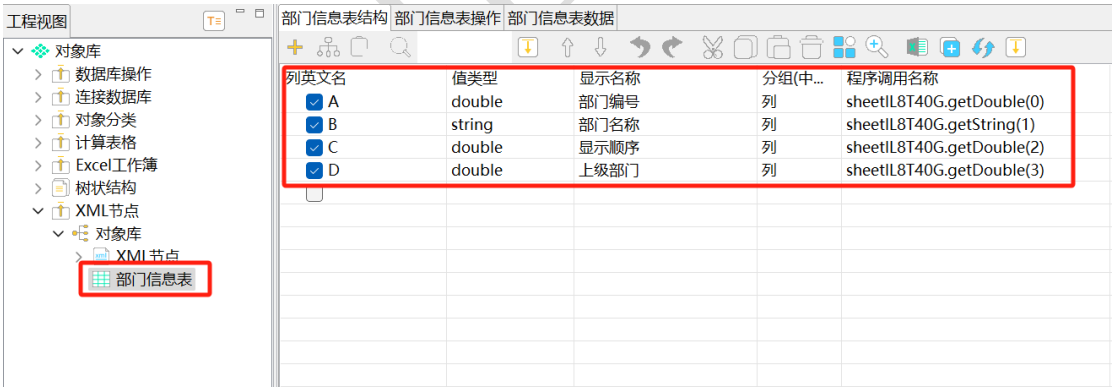
右键“部门信息表”选择“从 excel 中导入列信息”：



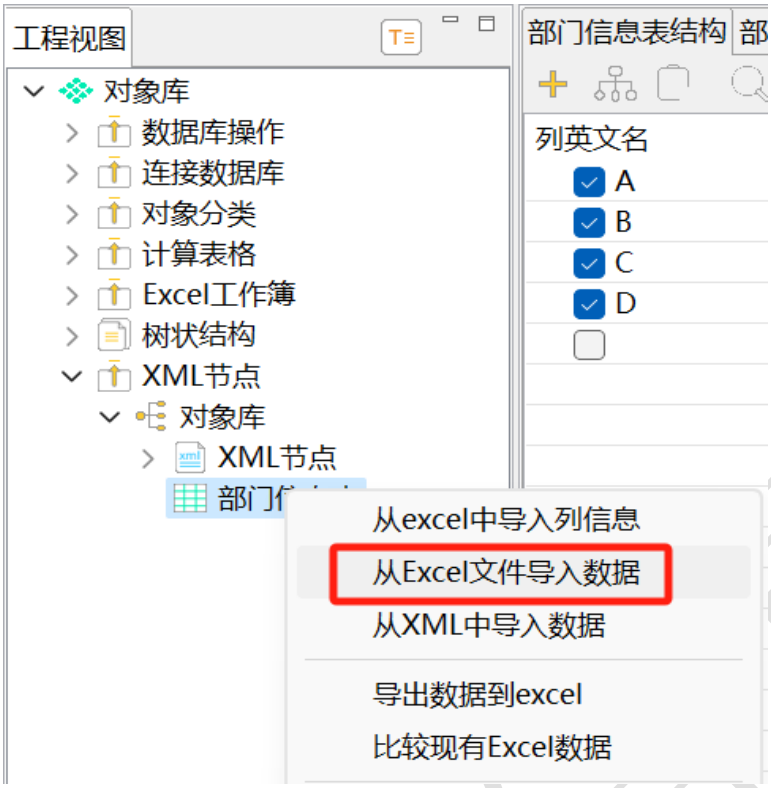
选择“部门信息.xls”，打开：



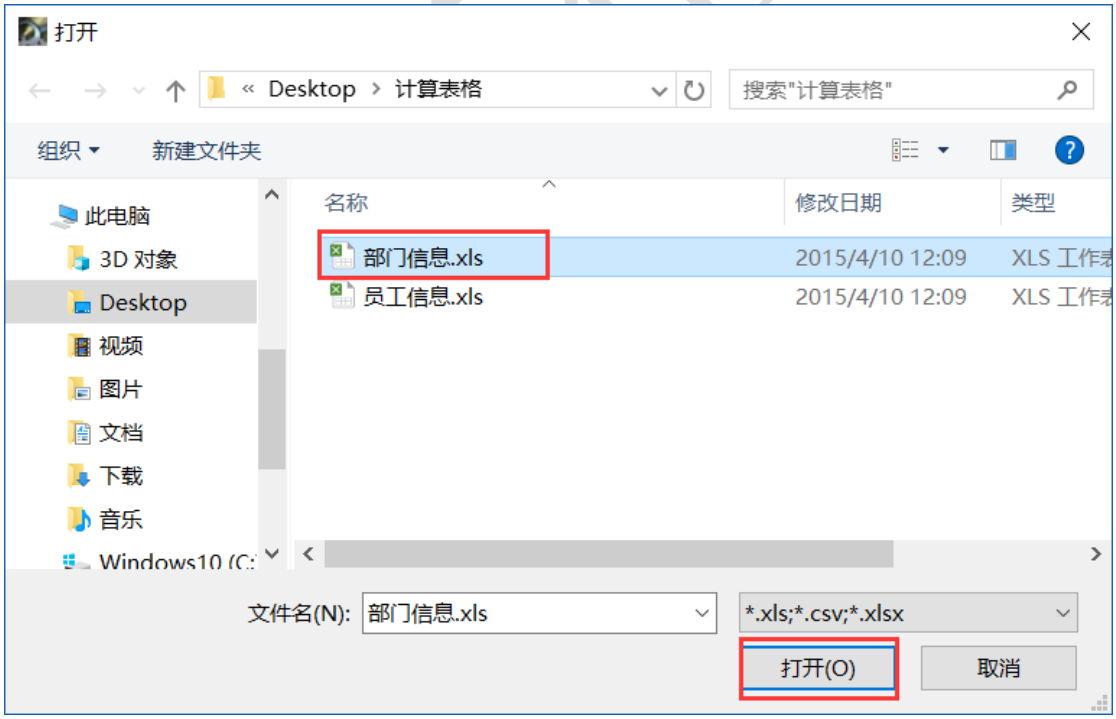
添加完成点击“部门信息表”查看结构：



有了列字段之后需要给表格添加数据，点击“部门信息表”选择“从 Excel 文件中导入数据”：



再次选择“部门信息.xls”，点击打开：



点击“部门信息表”，在编辑窗口的“部门信息表数据”中查看表数据：

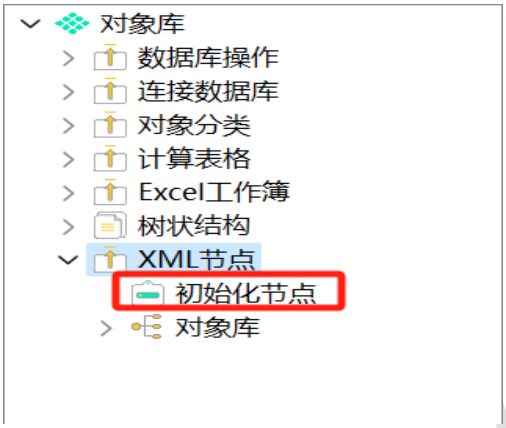
部门信息表结构 | 部门信息表操作 | 部门信息表数据

	部门编号	部门名称	显示顺序	上级部门
1	1	市场部	1	0
2	2	技术部	2	0
3	3	财务部	2	0

4．编写规则

(1) 新建“初始化节点”规则

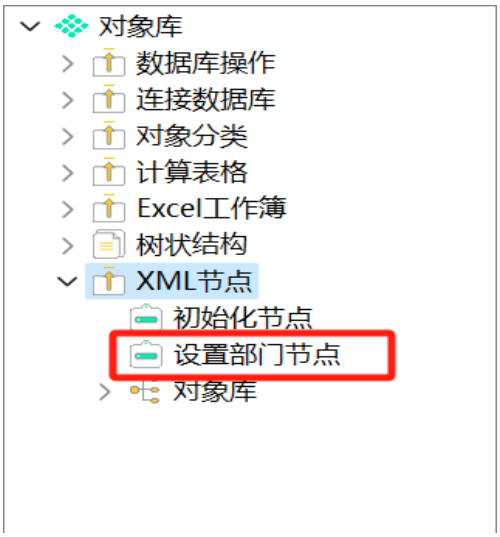
点击“XML 节点”规则包，右键添加规则，并命名为“初始化节点”：



在编辑窗口添加动作。执行方法→xml 节点→方法：

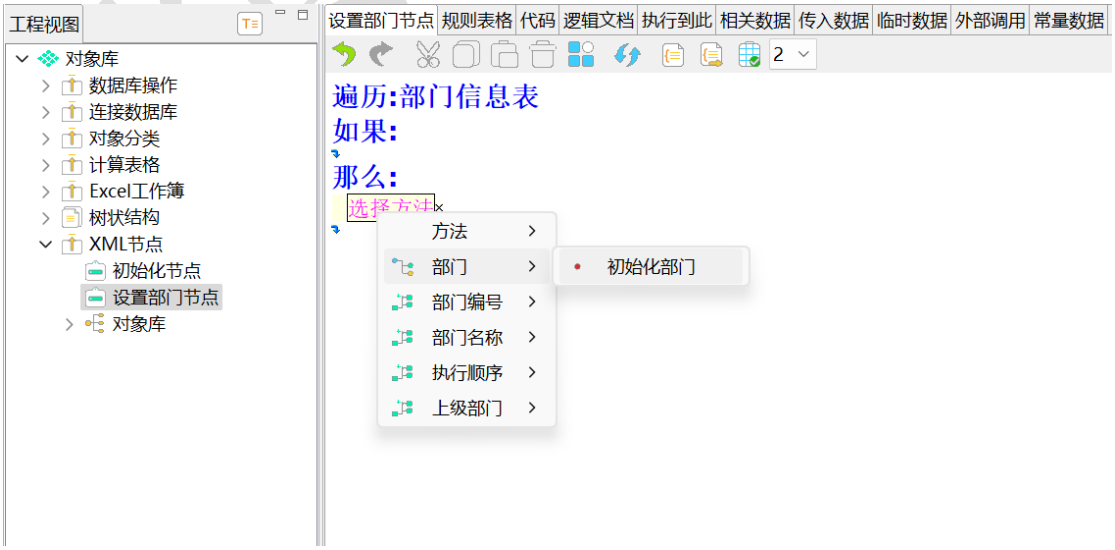
(2) 新建“设置部门节点”规则

右键“XML 节点”规则包，添加规则，命名为“设置部门节点”：

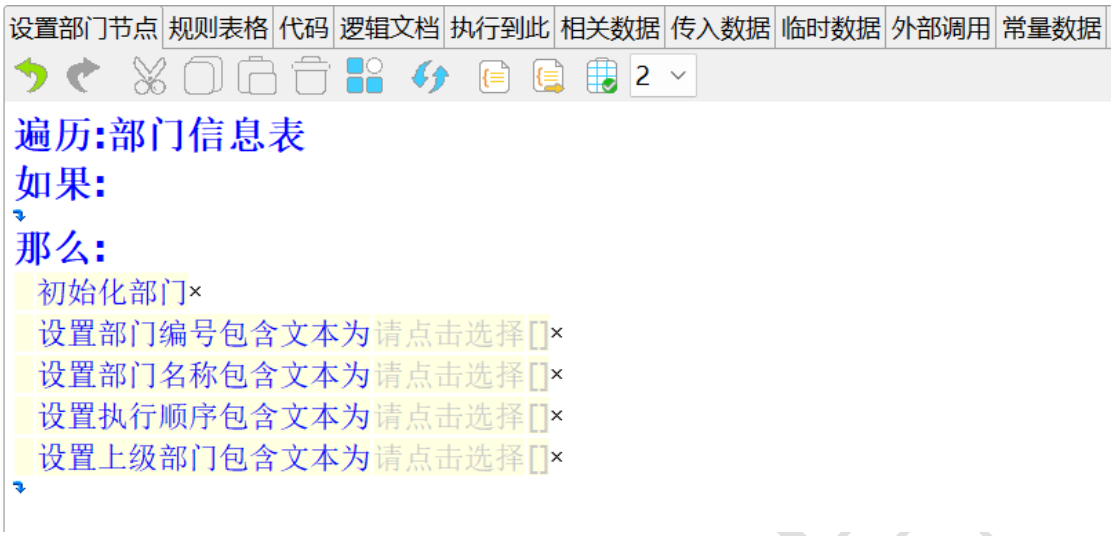


在“设置部门节点”属性窗口中设置遍历表格。将“遍历表格”对应的属性值勾选，再出现的行中选择表格“部门信息表”：

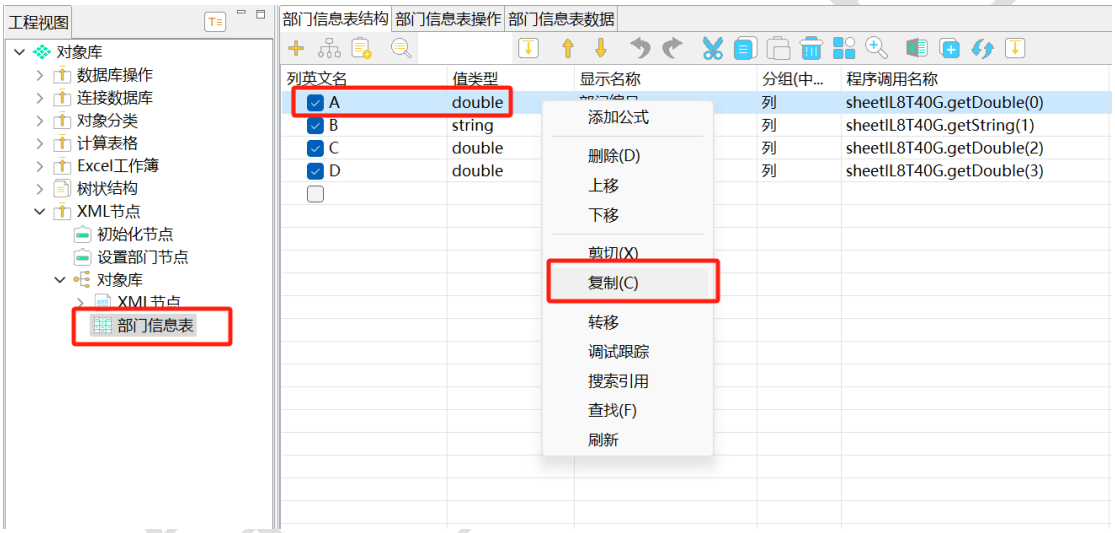
在编辑窗口中点击添加动作，选择 执行方法→部门：



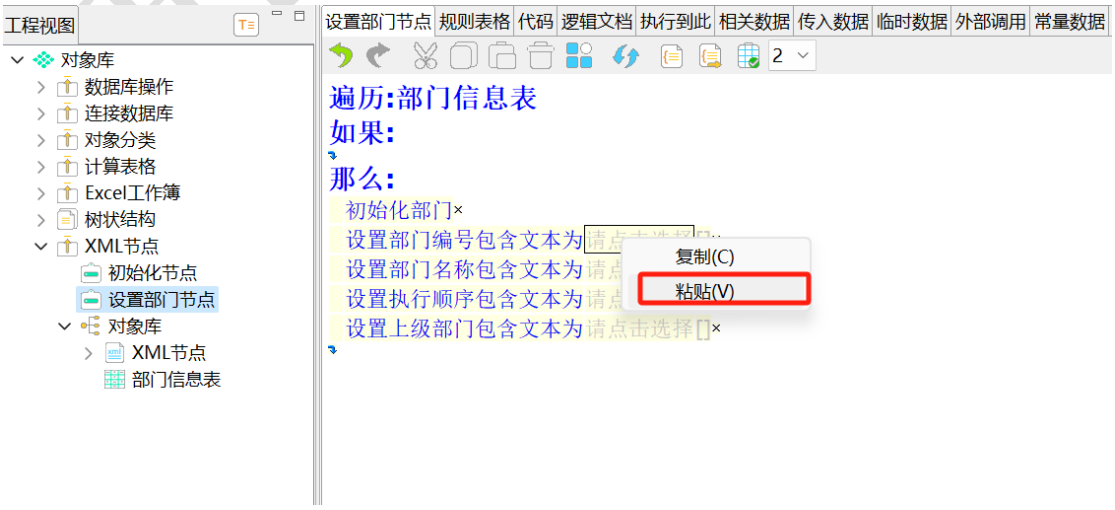
点击添加动作，依次添加下列方法：



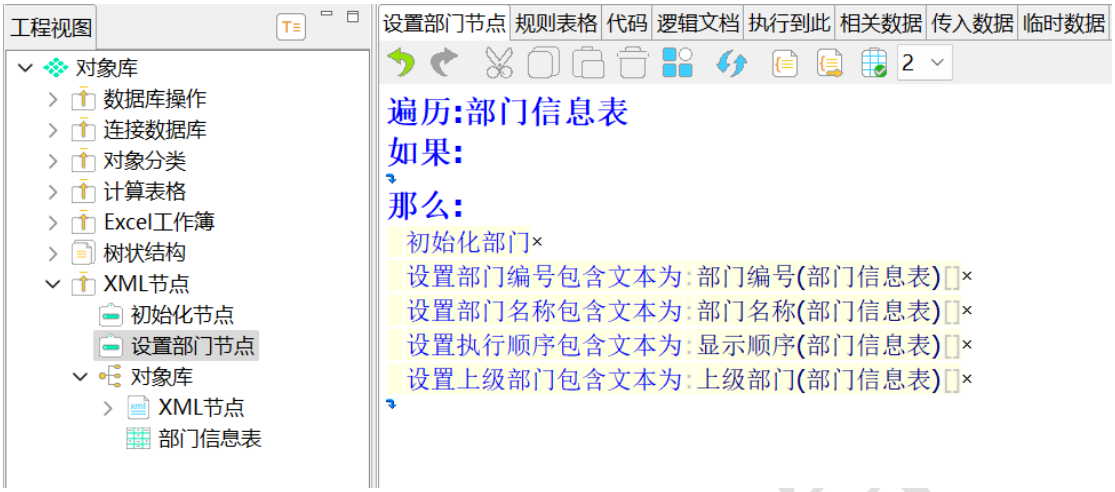
设置上述方法的值，点击“部门信息表”，在表结构中复制“部门编号”字段：



粘贴到“设置部门节点”的规则对应的值：

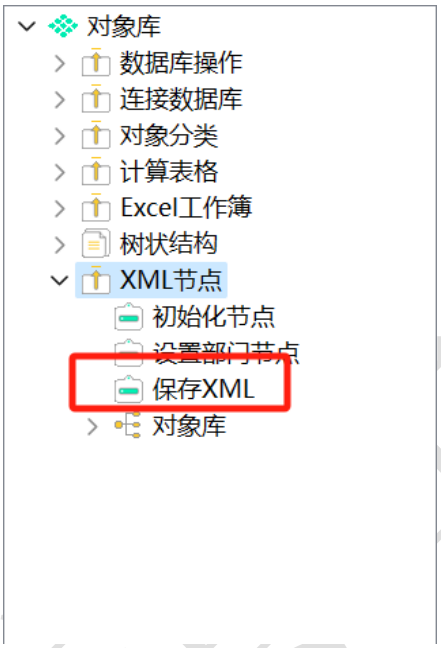


根据上述步骤，将其余的三个字段进行赋值：

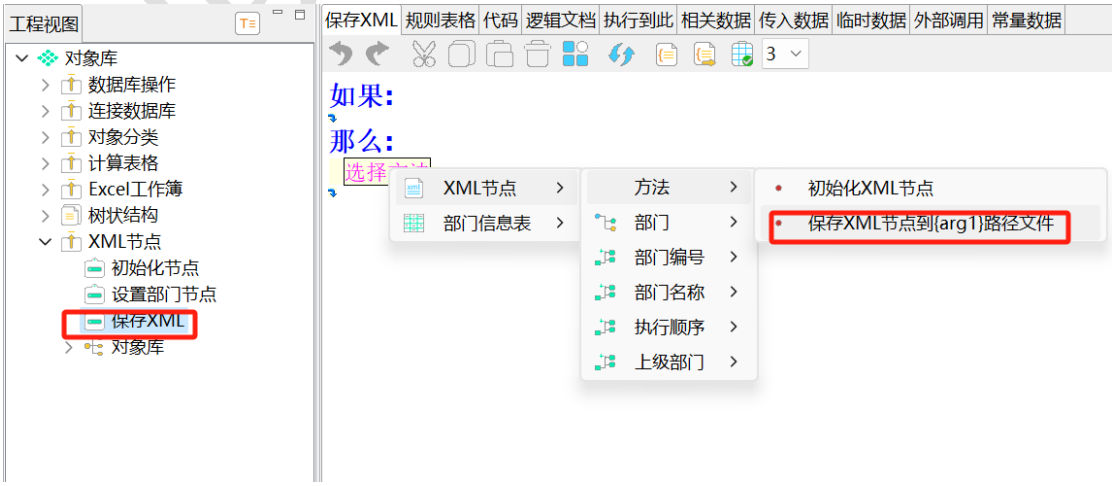


(3) 新建“保存 xml”规则

右键“XML 节点”规则包，添加规则，并命名为“保存 xml”：



编辑窗口中添加动作，执行方法→xml 节点→方法：

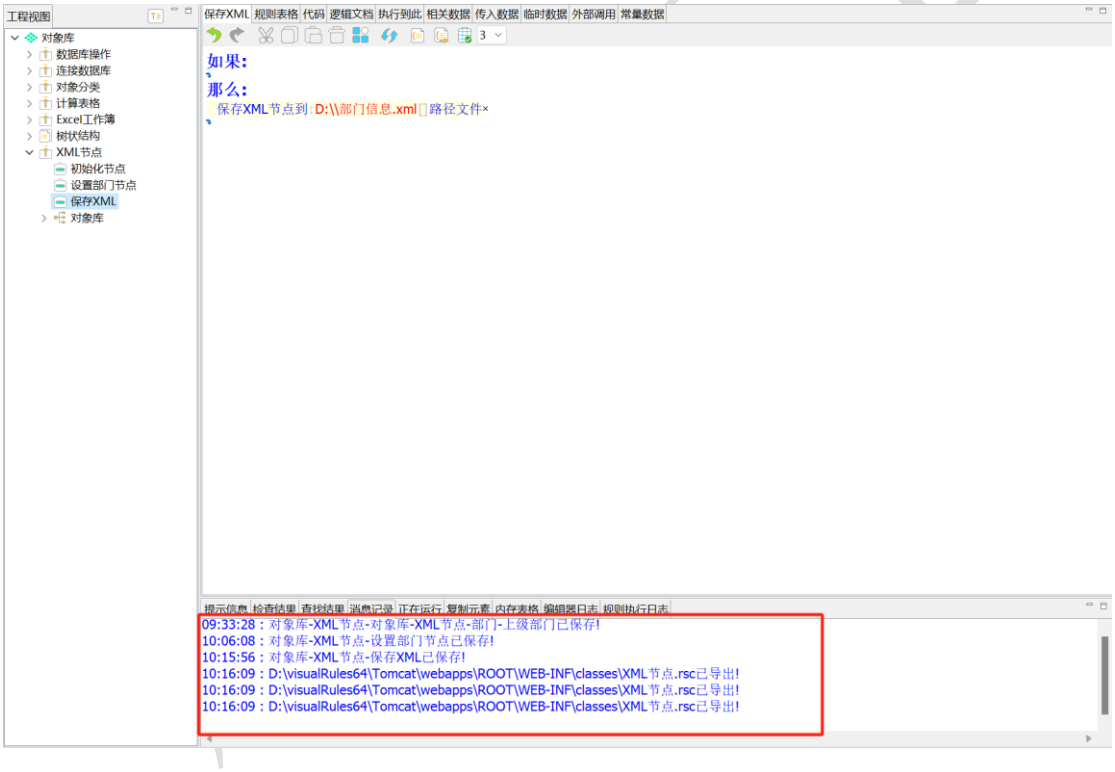


点击“请点击选择”输入常量值，点击“空”，在弹出框中填写 xml 文件生成路径：

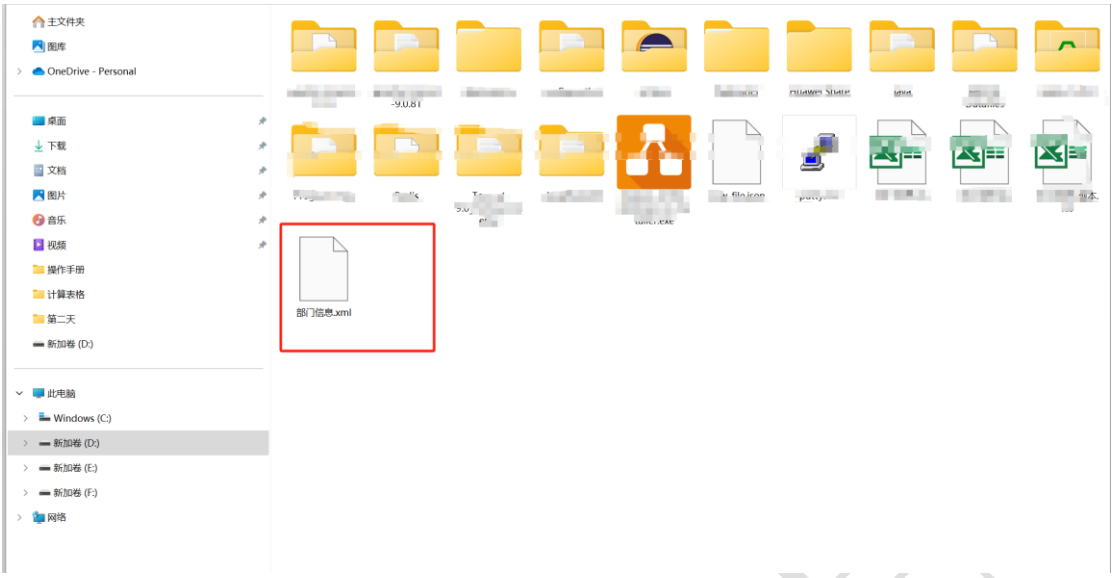
所有都完成之后点击“”保存并编译。

5．测试

点击“xml 节点”规则包，选择“测试”，点击“”执行：



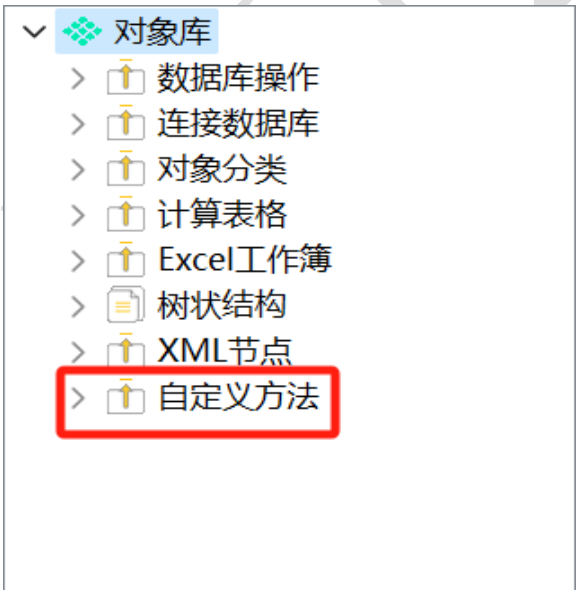
打开刚刚填写的路径，查看生成文件：



2.3.7. 自定义方法

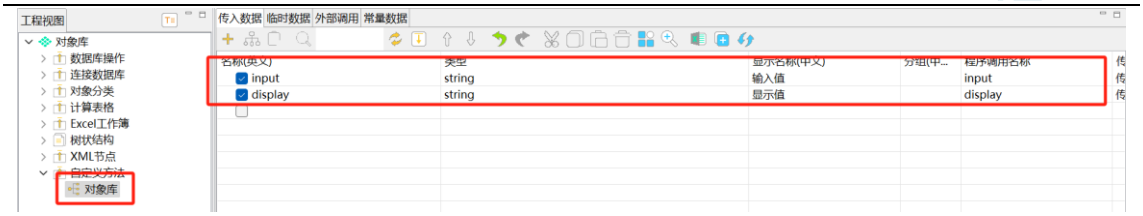
1. 创建规则包

在“对象库”工程下新建名为“自定义方法”的规则：



2. 定义变量

打开“自定义方法”规则包，在对象库传入数据中定义如下变量：



3．创建自定义方法

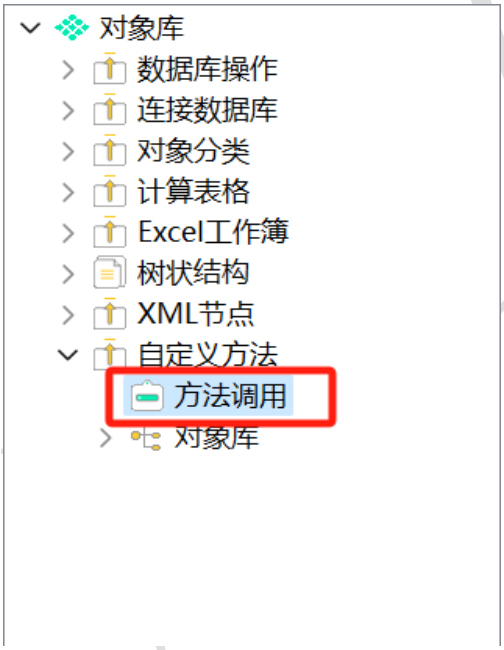
右键对象库，添加自定义方法：

打开自定义方法，引入以下代码：

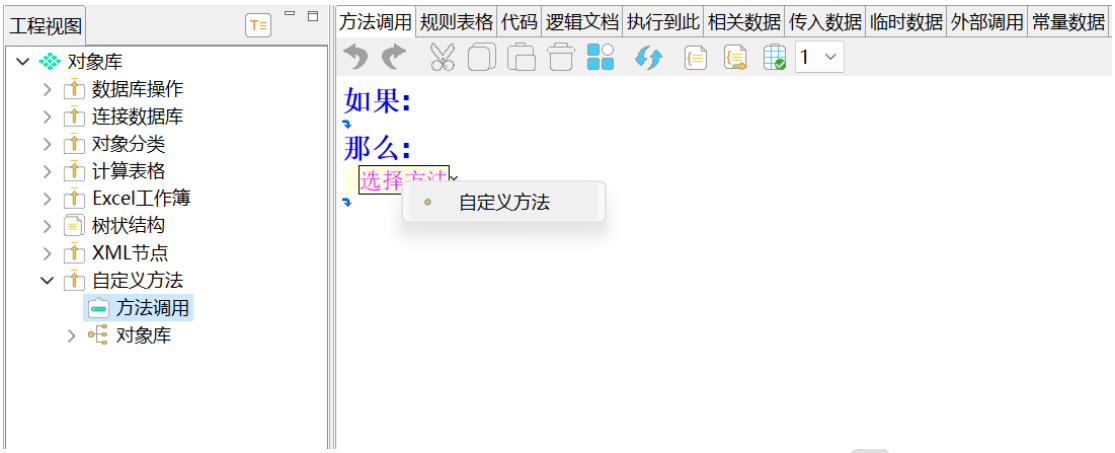
```
public String function(){
    if(input==""){
        return display="hello world";
    }else{
        return display= "hello"+input;
    }
}
```

4．自定义方法调用

点击“自定义方法”规则包，右键新建规则“方法调用”：



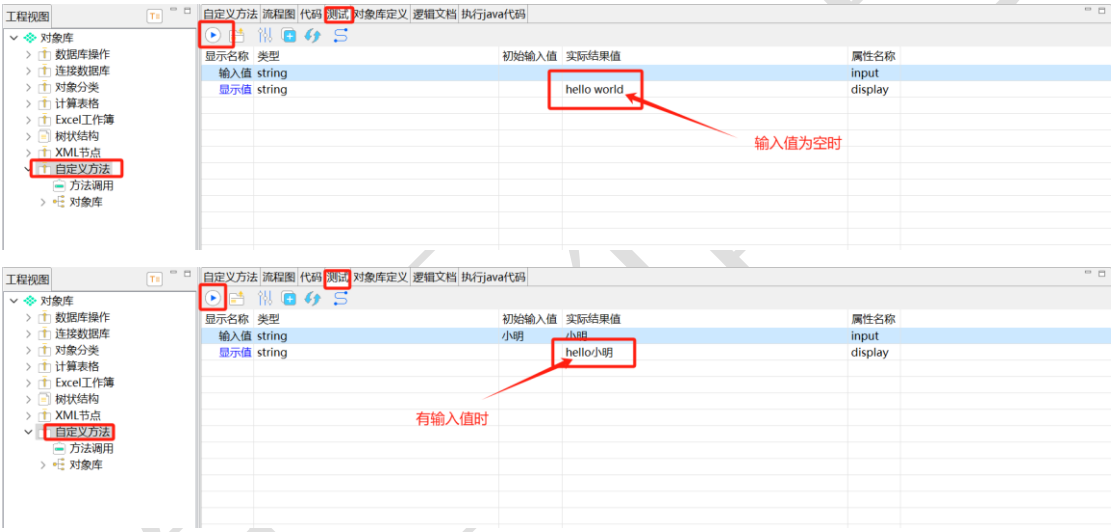
编辑窗口中添加动作，点击选择执行方法→自定义方法：



保存并编译。

5．测试

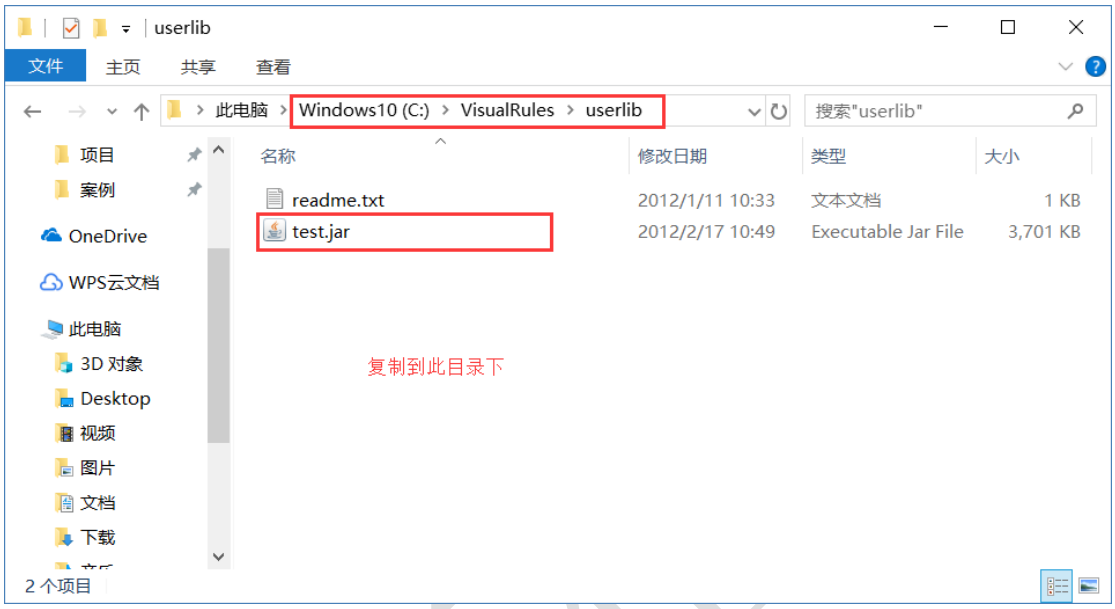
在“自定义方法”规则包的测试中，点击测试：



2.3.8. 接口实例

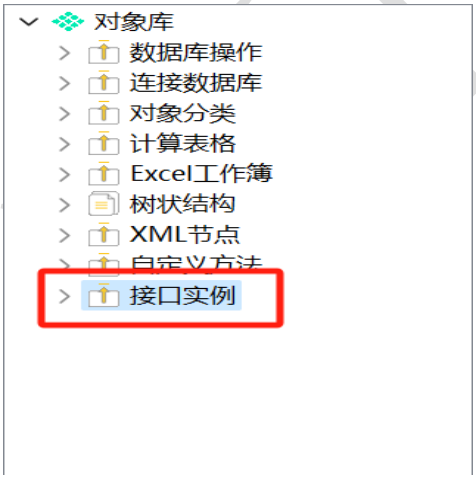
1. 导入 jar 包

将 test.jar 包放入到规则引擎安装目录 VisualRules\userlib 下，如果规则引擎是打开的，则需要重启。



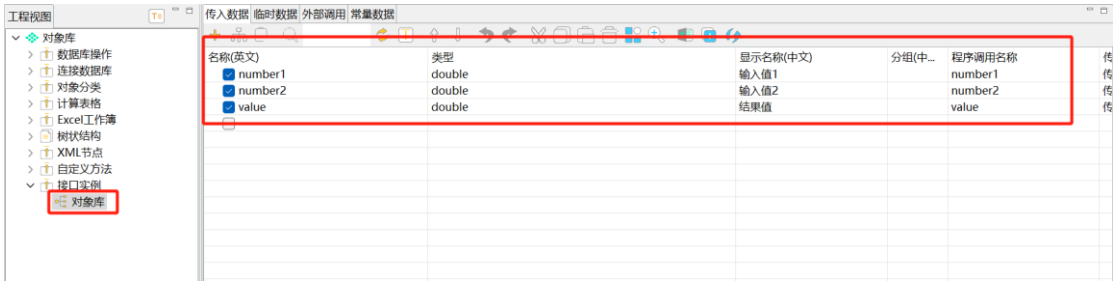
2. 创建规则

在“对象库”工程下，新建“接口实例”规则包：



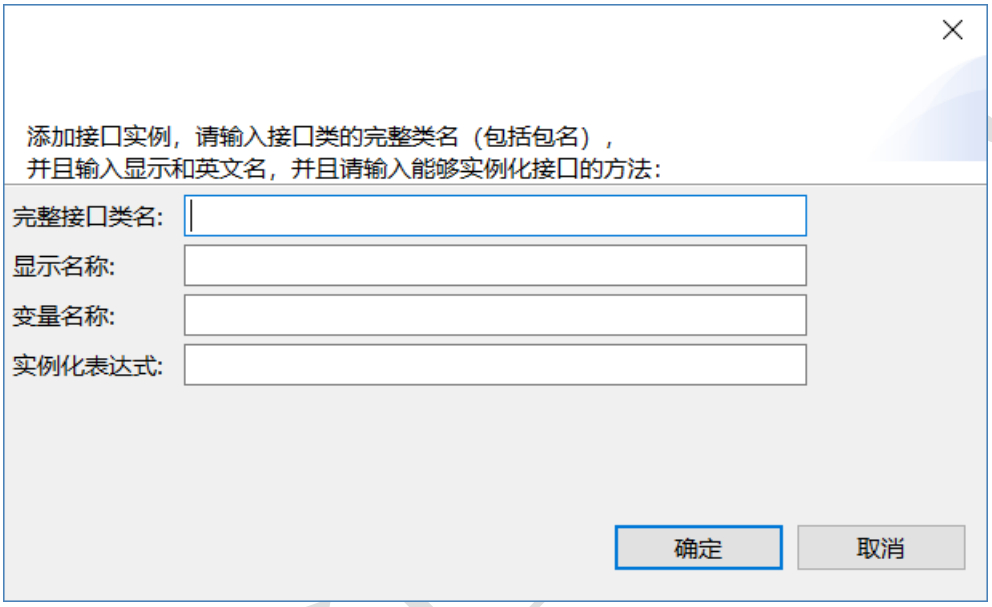
3. 定义变量

打开“接口实例”规则包，在对象库传入数据中添加如下变量：

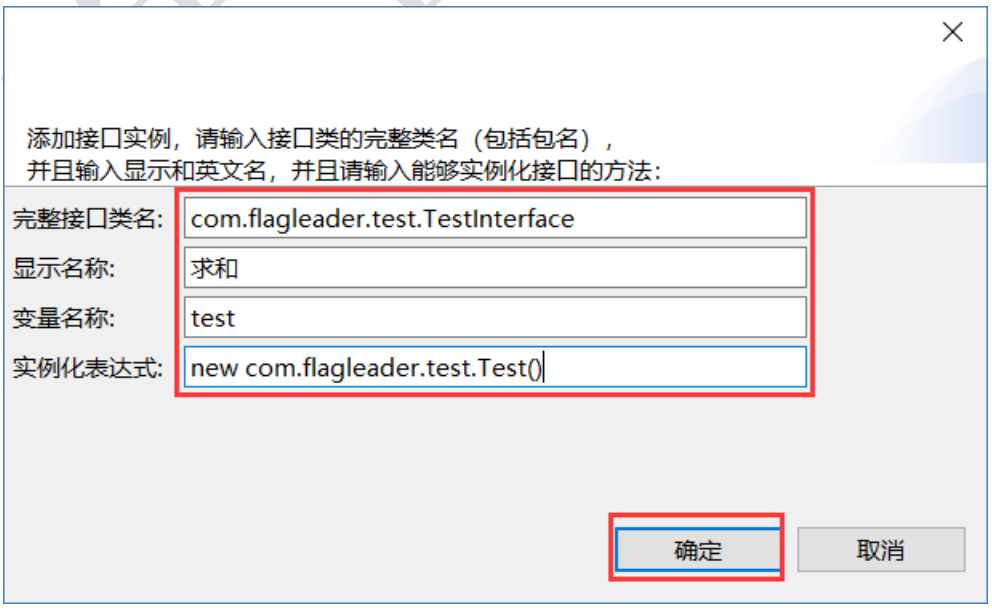


4．添加接口实例

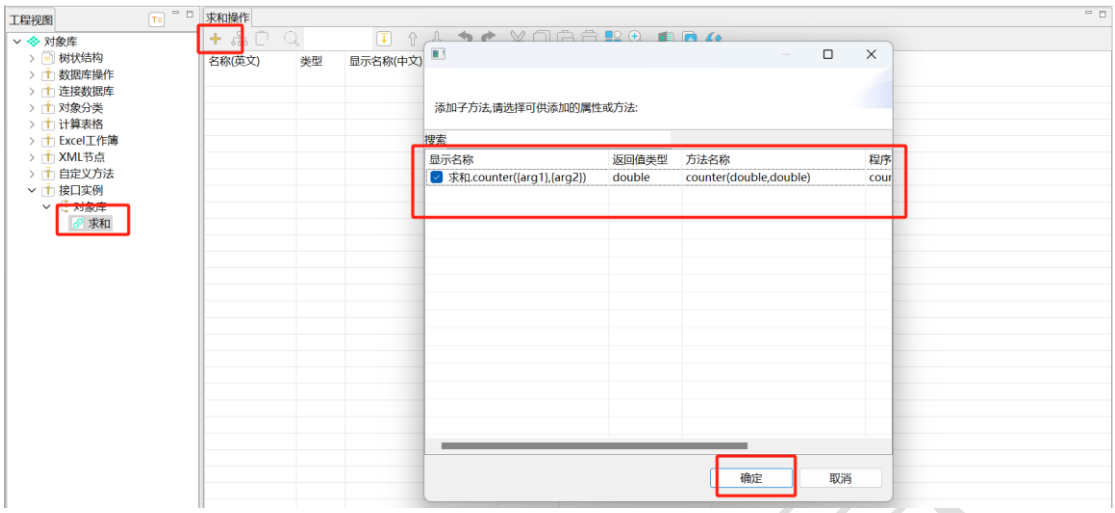
点击对象库，右键添加接口实例，弹出如下界面：



在弹出框中输入以下信息：

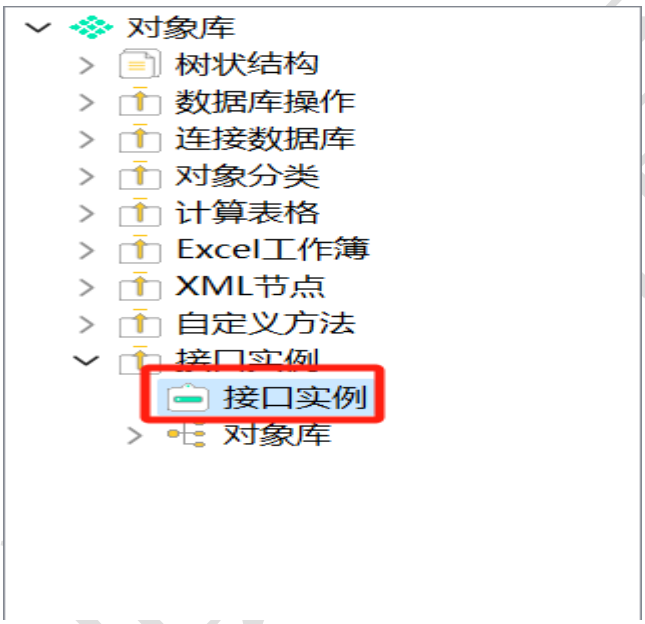


在接口“求和”编辑窗口中，点击“ ”添加方法，在弹出框中勾选，点击确定：

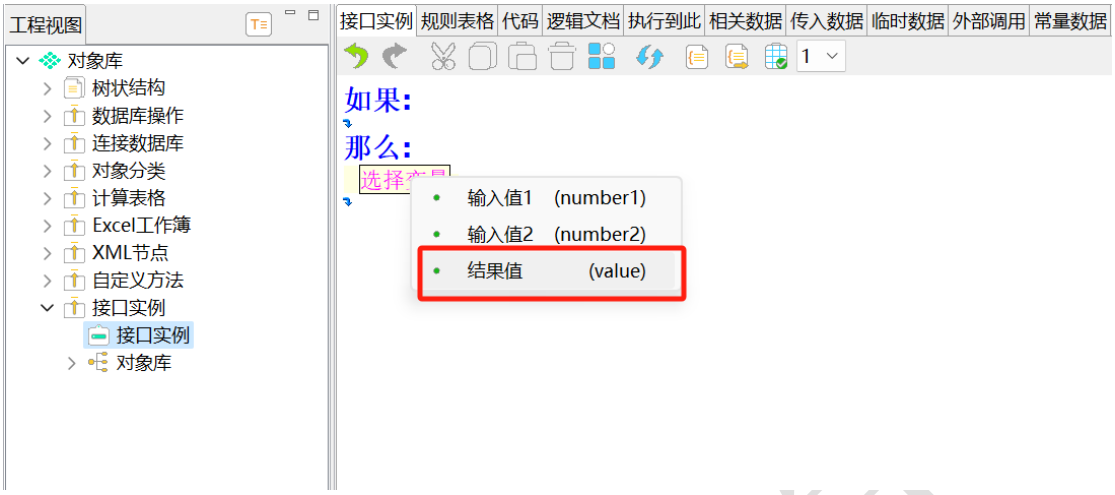


5．新建规则

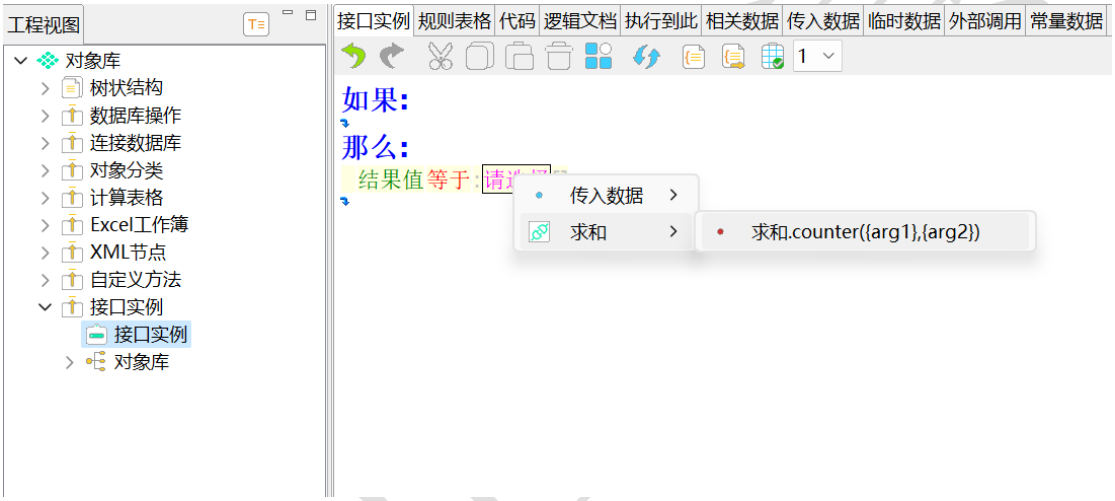
点击“接口实例”规则包，右键新建规则，并命名为“接口实例”：



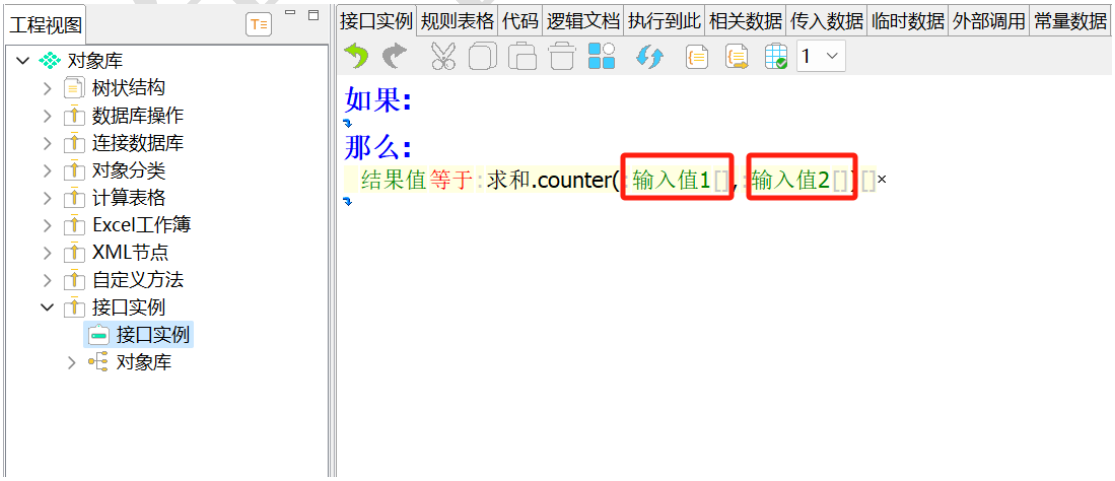
编辑窗口中添加动作，变量赋值→结果值 1：



给“结果值”赋值，选择选择值→求和：



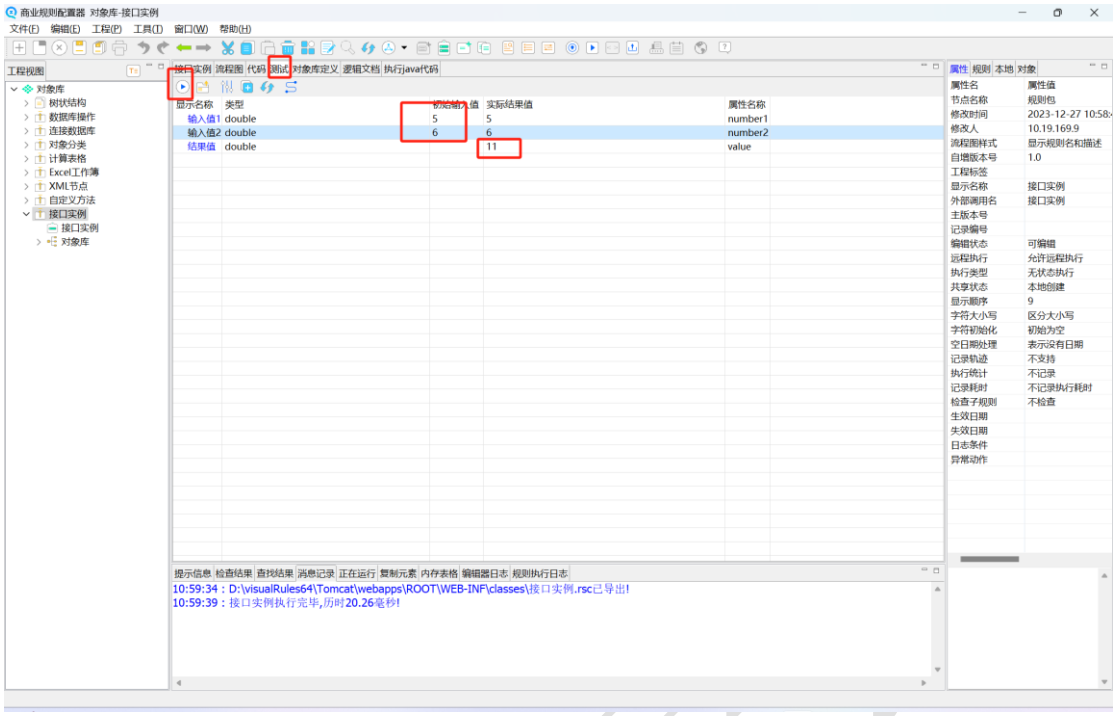
其中{arg1}选择“输入值 1”，{arg2}选择“输入值 2”：



保存并编译。

6. 测试

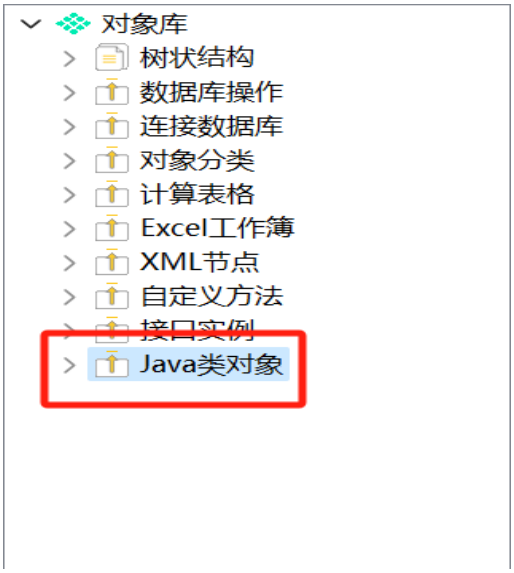
点击“接口实例”规则包，在测试中输入初始值，进行测试：



2.3.9. Java 类对象

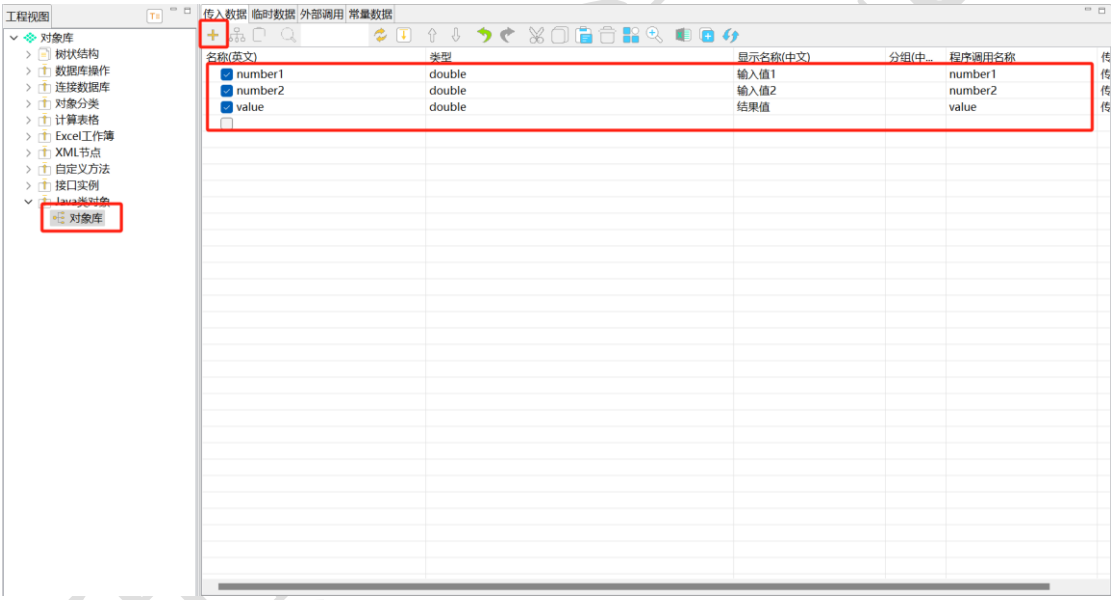
1. 创建规则包

在“对象库”工程下，新建名为“Java 类对象”规则包：



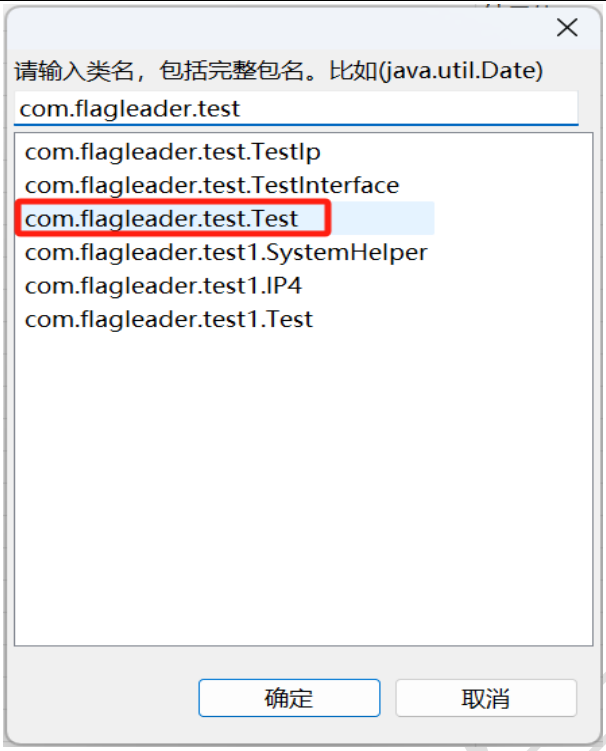
2．定义变量

打开“Java 对象库”规则包，在对象库中定义变量，如下图：

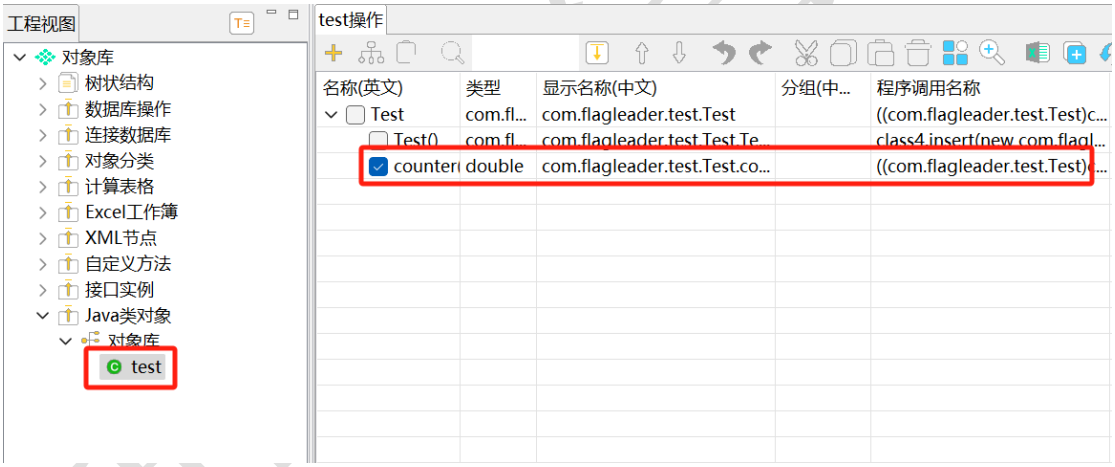


3．添加 Java 类对象

点击对象库，右键添加 Java 类对象，在弹出框中搜索 com.flagleader.test.Test：

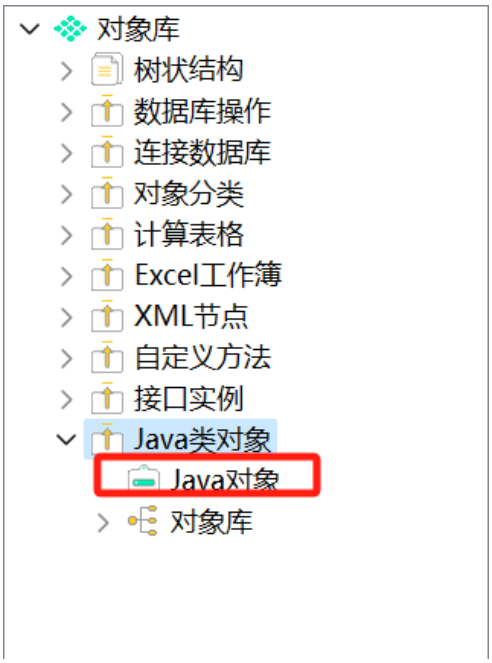


点击确定，在添加的 Java 类对象中，勾选 counter 子方法：

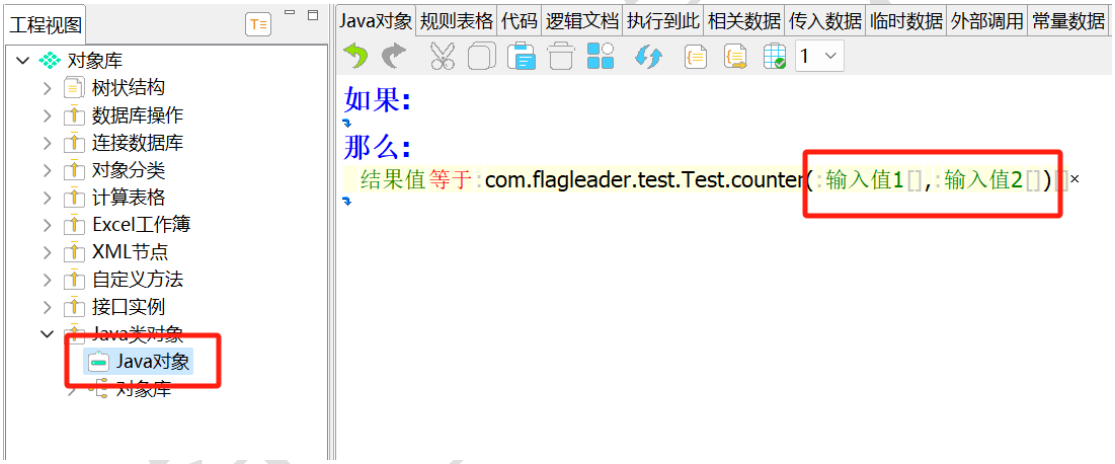


4. 新建规则

在“Java 类对象”规则包下，右键添加规则，命名为“Java 对象”：



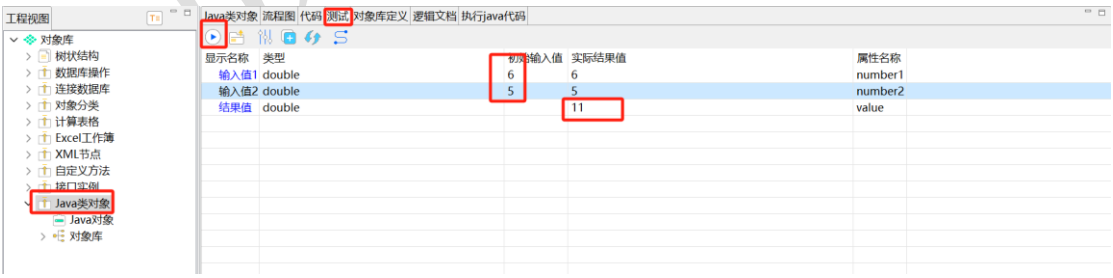
根据上述“接口实例”中的规则配置步骤，完成如下配置：



完成后保存并编译。

5. **测试**

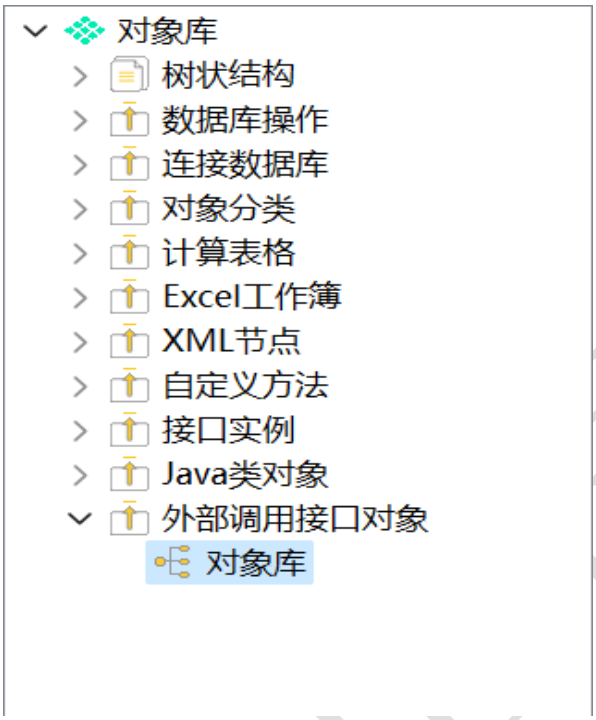
在“Java 类对象”测试窗口中，输入初始值测试：



2.3.10. 外部调用接口对象

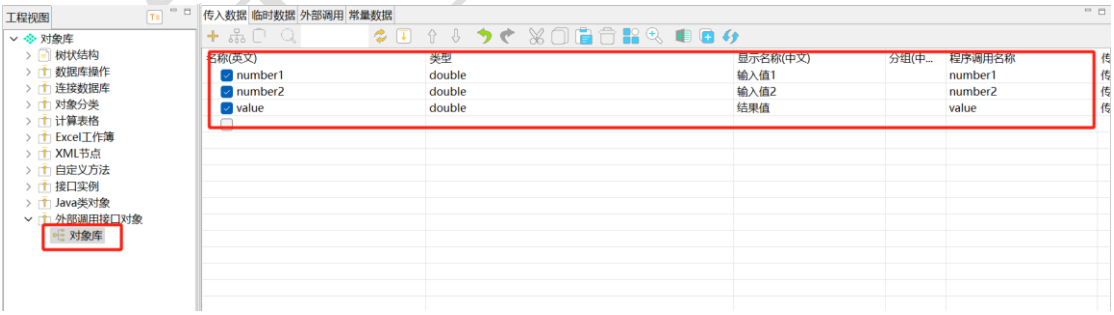
1．创建规则包

在“对象库”工程下新建规则，并命名为“外部调用接口对象”：



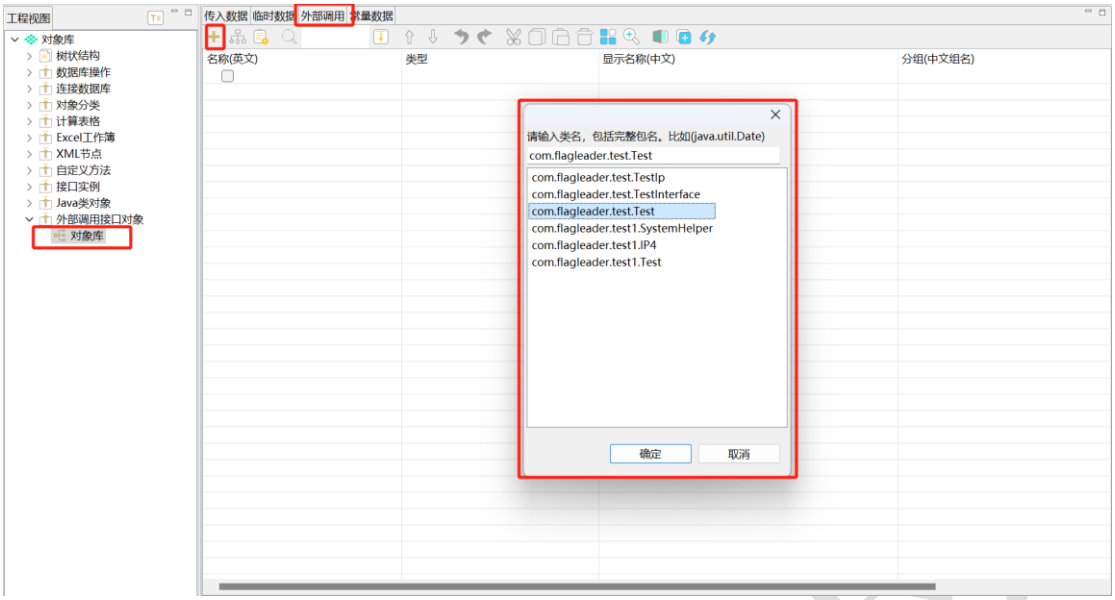
2．定义变量

在对象库传入数据中添加以下变量：

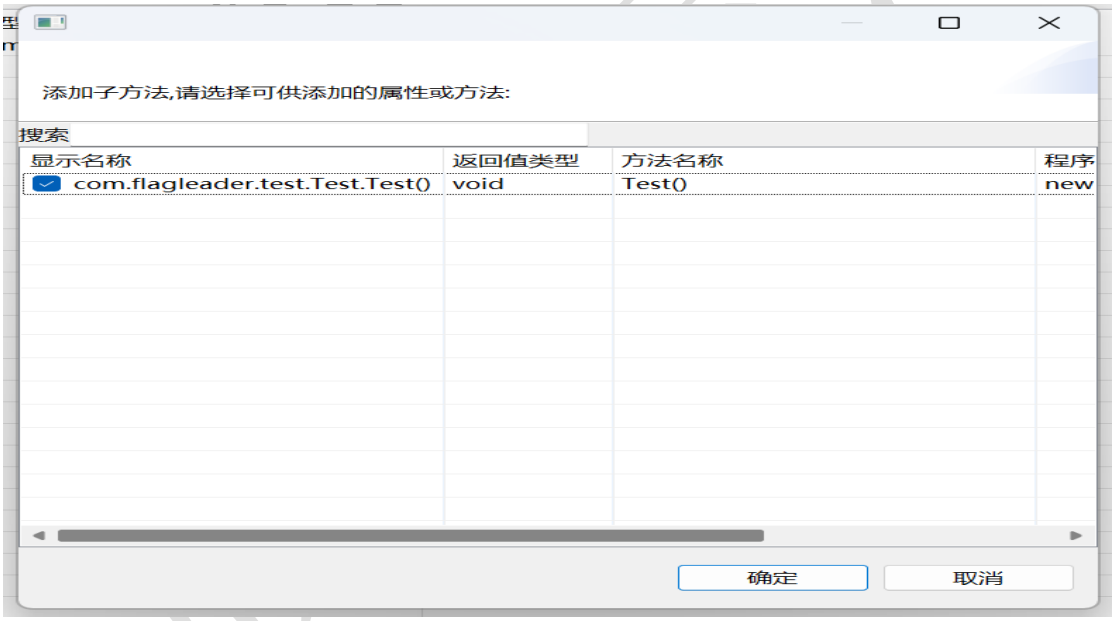


3．添加外部调用

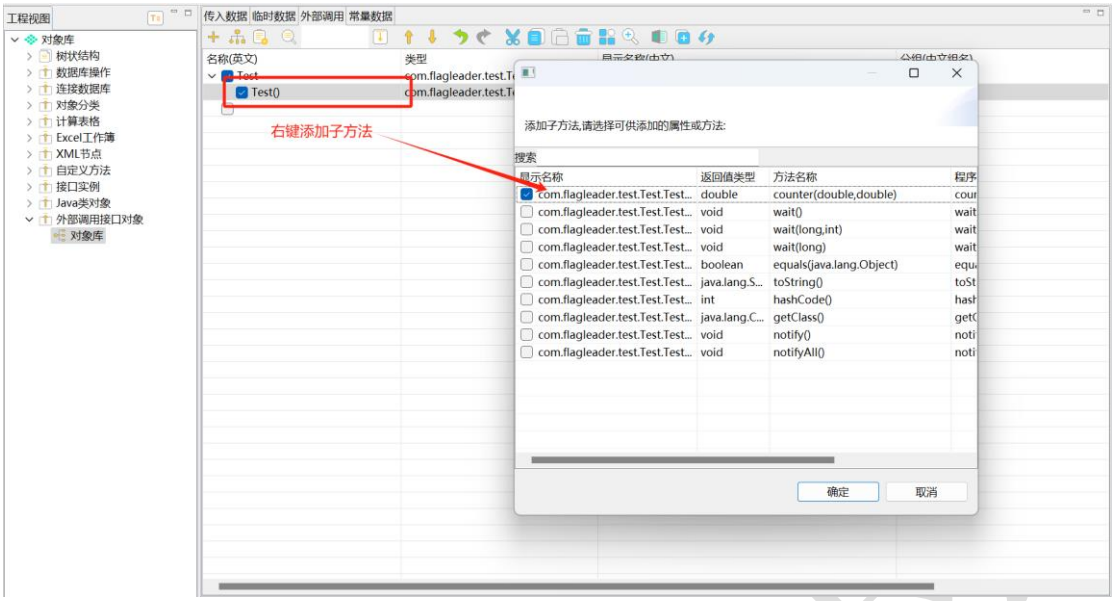
切换至外部调用选项卡，点击工具栏中的“”，添加方法，在弹出页面中输入com.flagleader.test.Test：



在弹出框中选择子方法:

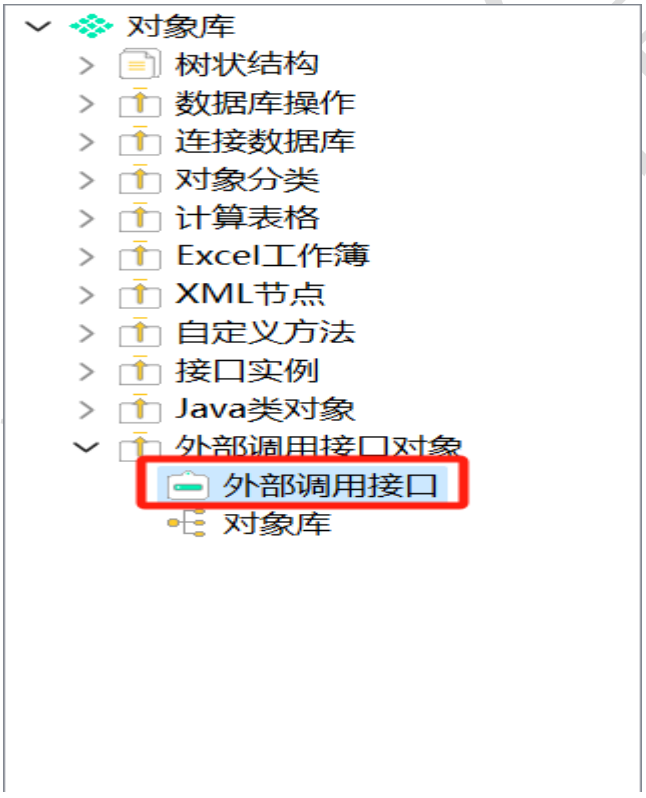


点击刚刚添加的方法，右键添加子方法，在弹出框中勾选改方法：

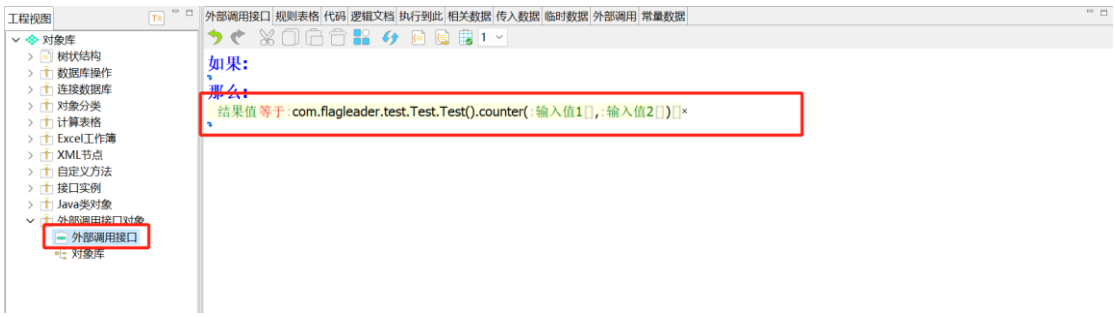


4．新建规则

在“外部调用接口对象”下添加规则，命名为“外部调用接口”：



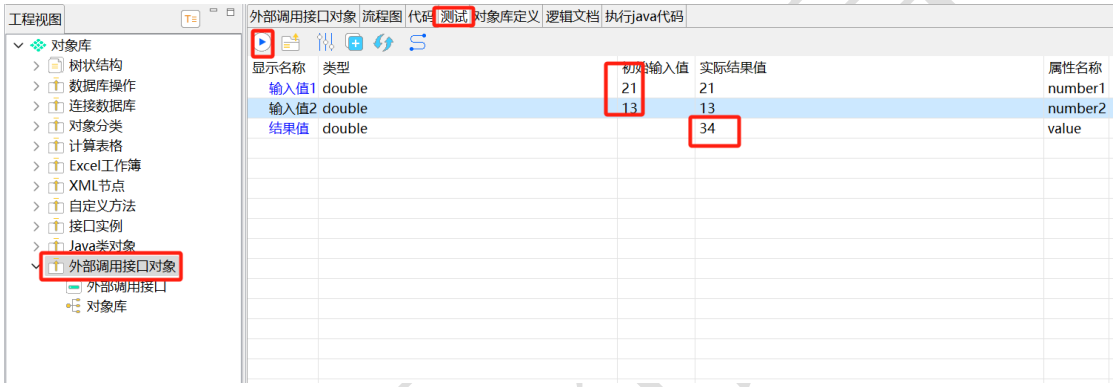
参照“接口实例”完成规则配置：



保存并编译。

5．测试

点击“外部调用接口对象”，在编辑窗口中切换至“测试”，输入初始值完成测试：



2.3.11. 扩展函数

在规则引擎中除了原有的方法外，还支持用户自定义方法。用户可将自定义方法放在 xml 文件中，然后将 xml 文件放在工具安装目录下的 classes 目录下，在规则中通过全局方法调用 xml 文件中的自定义方法。xml 文件格式如下图所示：

```
<?xml version="1.0" encoding="utf-8"?>
<MethodInfos displayName="java">
    <GlobalMethod funcName="round(double)" disName="四舍五入{double}"
        returnType="long" typeName="数值处理">
        <funcBody>MathUtil.round({double})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="round(float)" disName="四舍五入{float}"
        returnType="int" typeName="数值处理">
        <funcBody>MathUtil.round({float})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="abs(double)" disName="取{double}的绝对值"
```

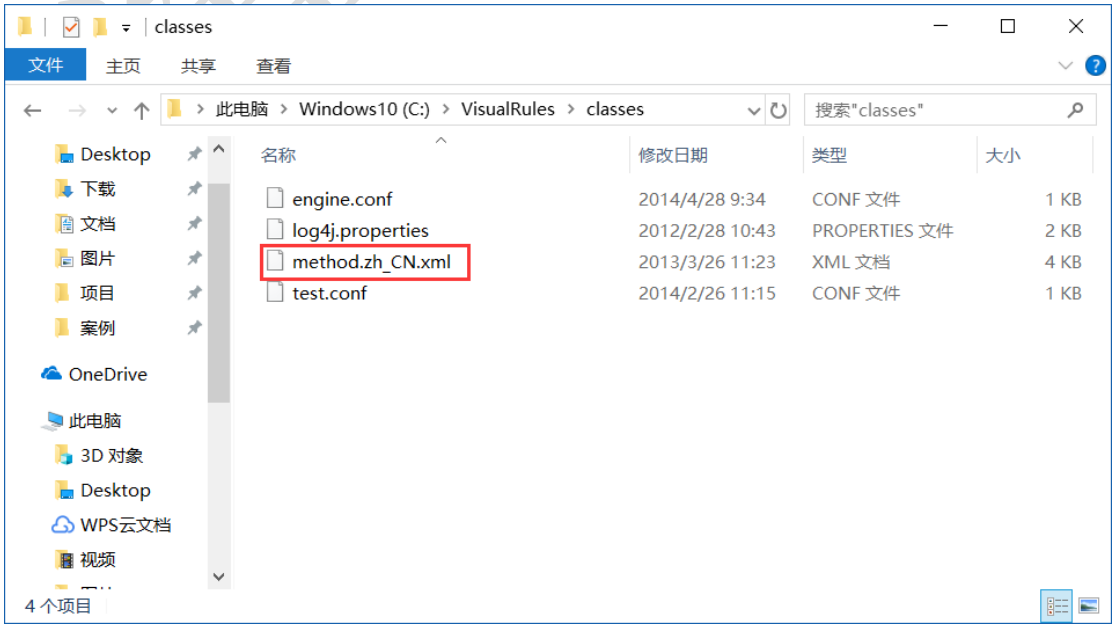
```
        returnType="double" typeName="数值处理">
        <funcBody>MathUtil.abs({double})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="max(double,double)" disName="取{double}和{double}
中的最大值"
        returnType="double" typeName="数值比较">
        <funcBody>MathUtil.max({double},{double})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="min(double,double)" disName="取{double}和{double}
中的最小值"
        returnType="double" typeName="数值比较">
        <funcBody>MathUtil.min({double},{double})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="random()" disName="随机数值"
        returnType="double" typeName="数值处理">
        <funcBody>MathUtil.random()</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="parseInt(string)" disName="转换{string}为整形值"
        returnType="int" typeName="格式转换">

    <funcBody>com.flagleader.util.NumberUtil.parseInt({string})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="parseDouble(string)" disName="转换{string}为浮点
值"
        returnType="double" typeName="格式转换">
        <funcBody>com.flagleader.util.NumberUtil.parseDouble({string})
        </funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="DecimalUtil.add(double,double)"
        disName="精确累加{arg1}和{arg2}" returnType="double" typeName="精确计
算">
        <funcBody>DecimalUtil.add({double},{double})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="DecimalUtil.sub(double,double)"
        disName="精确相减{arg1}和{arg2}" returnType="double" typeName="精确计
算">
        <funcBody>DecimalUtil.sub({double},{double})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="DecimalUtil.mul(double,double)"
        disName="精确相乘{arg1}和{arg2}" returnType="double" typeName="精确计
算">
        <funcBody>DecimalUtil.mul({double},{double})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="DecimalUtil.div(double,double)"
```

```
        disName="精确相除{arg1}和{arg2}" returnType="double" typeName="精确计
算">
        <funcBody>DecimalUtil.div({double},{double})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="DecimalUtil.div(double,double,int)"
        disName="精确相除{arg1}和{arg2}，精度为{arg3}" returnType="double"
typeName="精确计算">
        <funcBody>DecimalUtil.div({double},{double},{int})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="DecimalUtil.round(double,int)"
        disName="四舍五入{arg1}，精度为{arg2}" returnType="double" typeName="
精确计算">
        <funcBody>DecimalUtil.round({double},{int})</funcBody>
    </GlobalMethod>
    <GlobalMethod funcName="DecimalUtil.round(double,int)"
        disName="四舍五入 1{arg1}，精度为1{arg2}" returnType="double"
typeName="精确计算 1">
        <funcBody>DecimalUtil.round({double},{int})</funcBody>
    </GlobalMethod>
</MethodInfos>
```

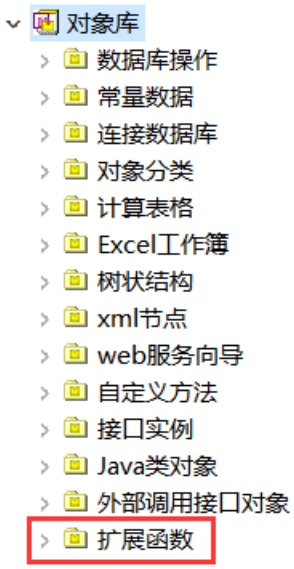
1. 新建 xml 文件

新建 xml 文件，将上面的 xml 格式的代码复制到其中，保存此 xml 文件并将该文件放在规则配置器安装目录下的 classes 目录中：



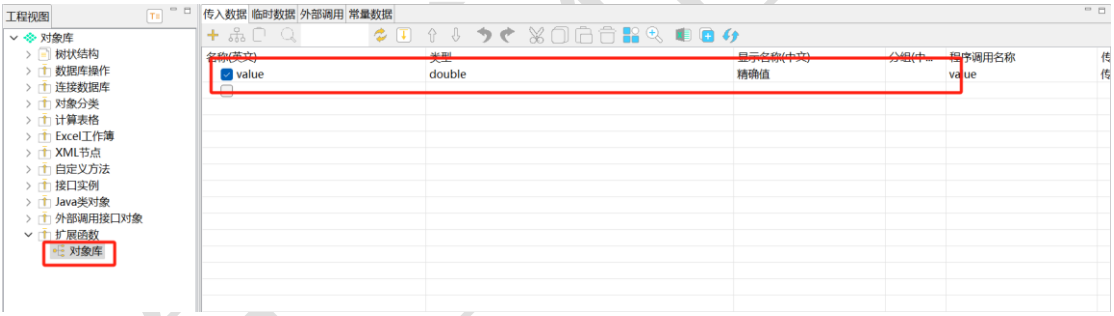
2．新建规则包扩展函数

在“对象库”工程下新建规则包“扩展函数”：



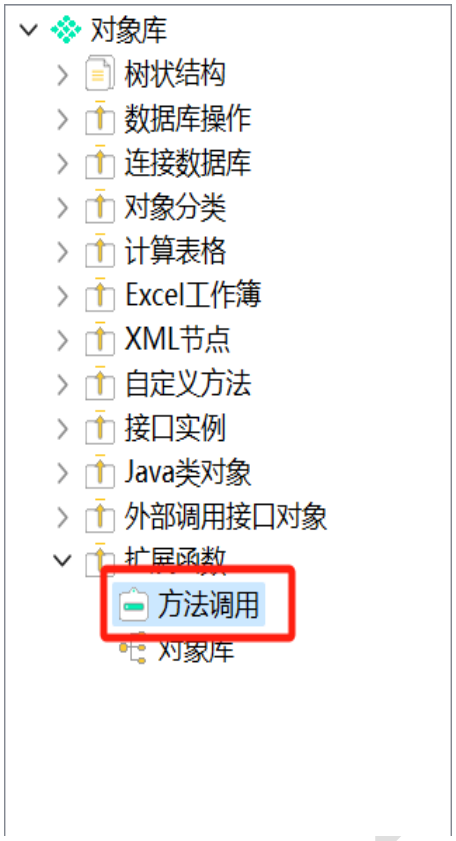
3．定义变量

在规则包“扩展函数”的对象库中添加一个 double 类型的变量“精确值（value）”：

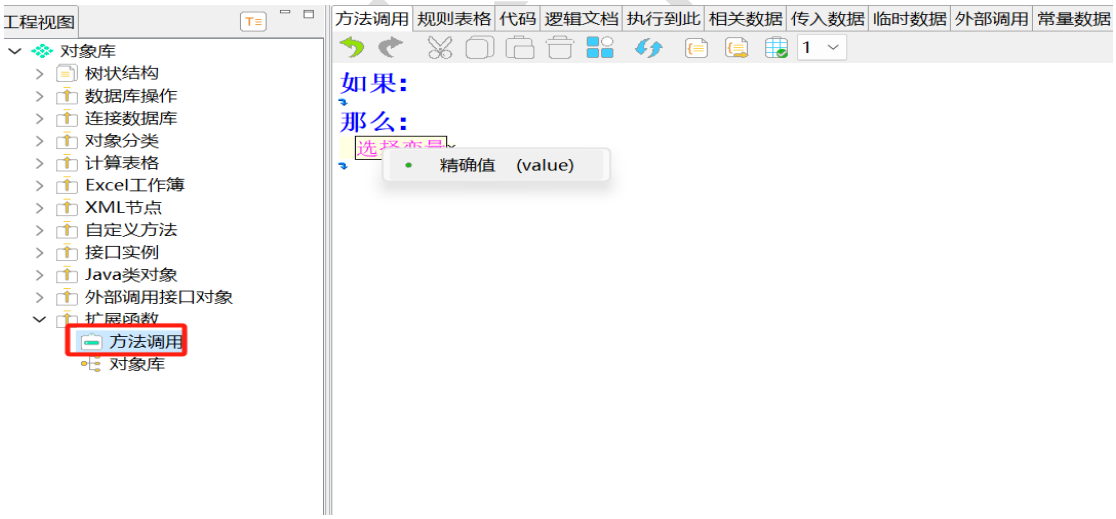


4．新建规则方法调用

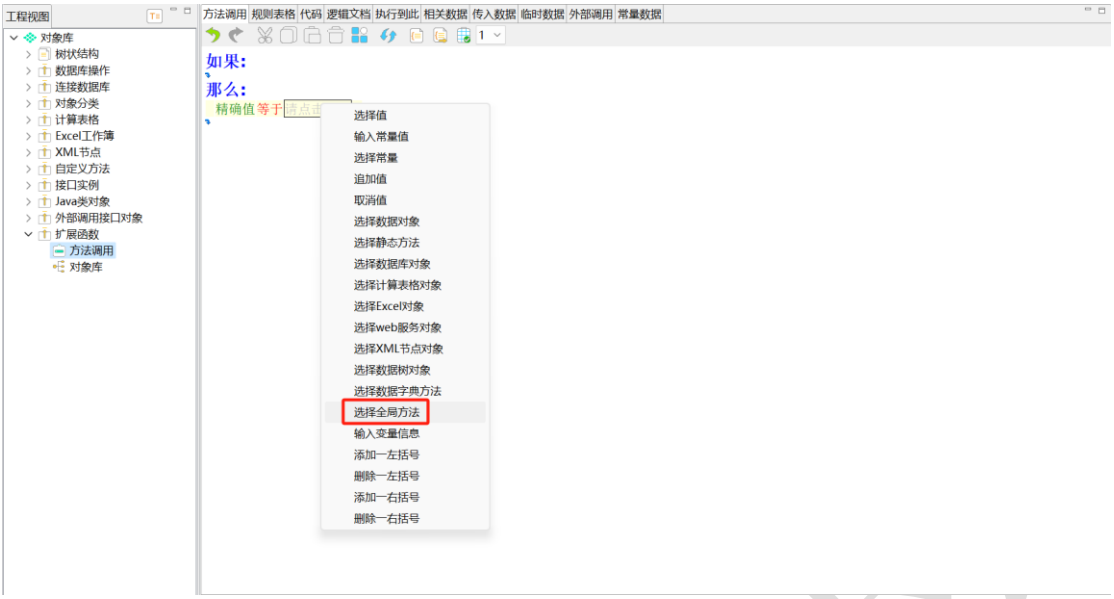
点击规则包“扩展函数”，选择菜单项“添加规则”,并将规则命名为“方法调用”，如下：



在规则“方法调用”的编辑窗口中点击“”添加动作，选择变量赋值→精确值：

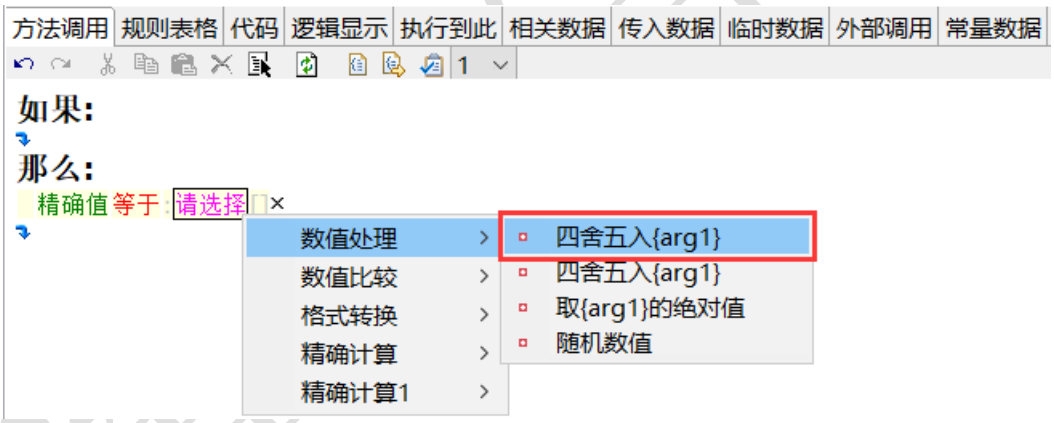


为变量“精确值”赋值，选择全局方法：

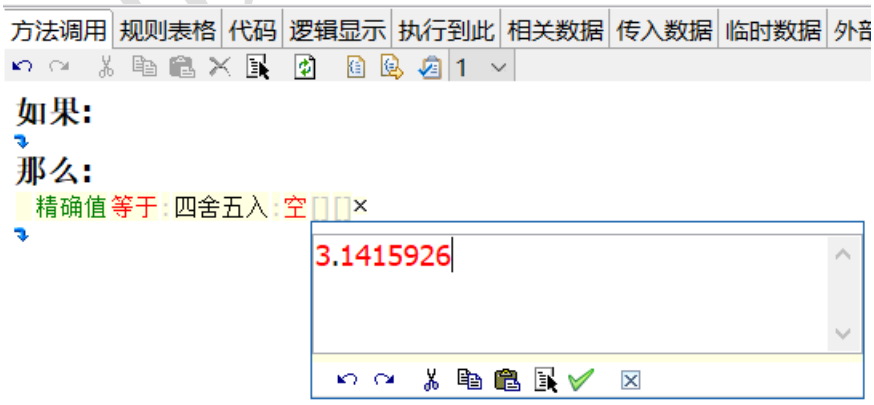


点击“请选择”出现的方法就是我们刚刚放入 VisualRules\classes 目录里 xml 中添加的方法，如点击“请选择”没反应请先将规则保存然后重启规则配置器。

选择其中的“四舍五入”：



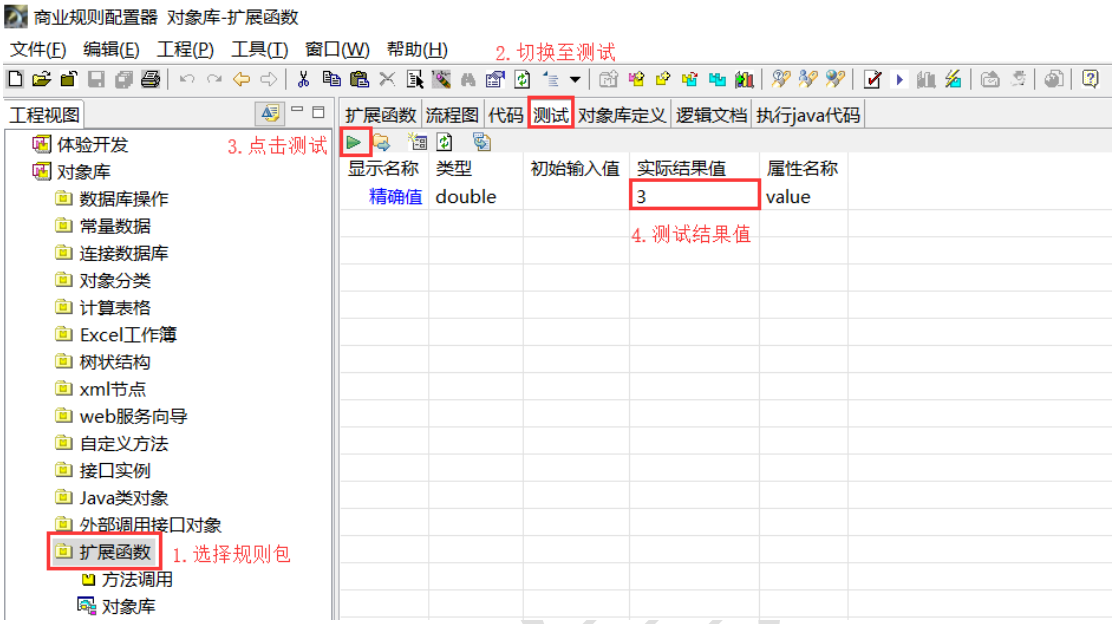
为{arg1}赋予常量值。点击“请点击选择”，选择输入常量值，点击空，在弹出框中输入常量值：



完成后保存并编译。

5．测试

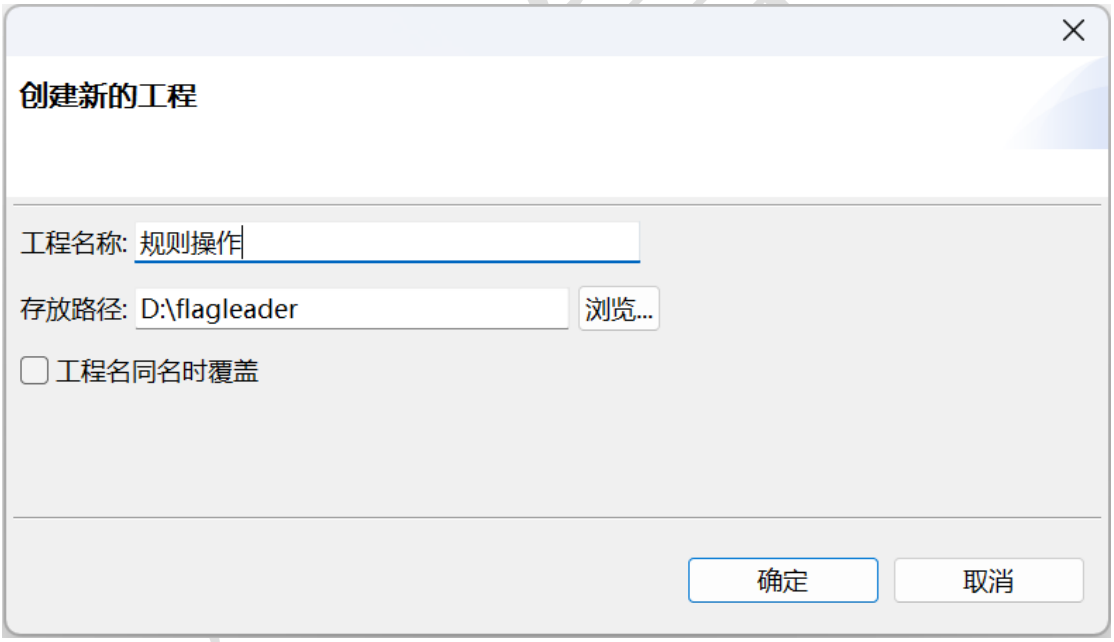
选择“扩展函数”规则包，在编辑窗口中进行测试：



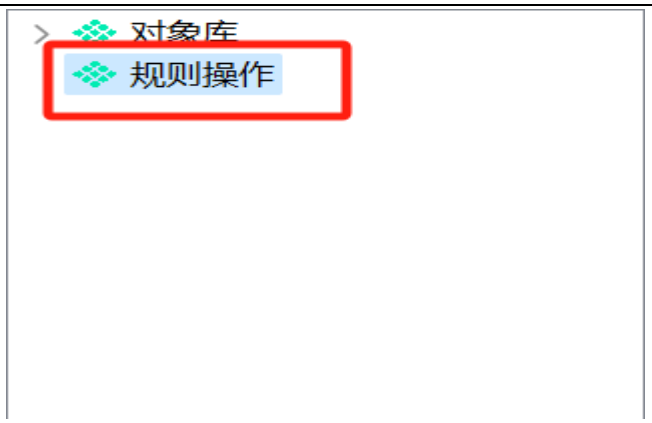
三、 规则操作

首先，需要创建规则工程。点击菜单栏上的“文件”，点击菜单项中的“新建规则工程”，如下图：

弹出对话框“创建新的工程”，在工程名称对应的文本框中输入“规则操作”。如果要更改存放路径点击“浏览...”，选择自己的存放路径，如下图:



点击确定，如下图：



规则工程创建完成。

3.1. 规则集

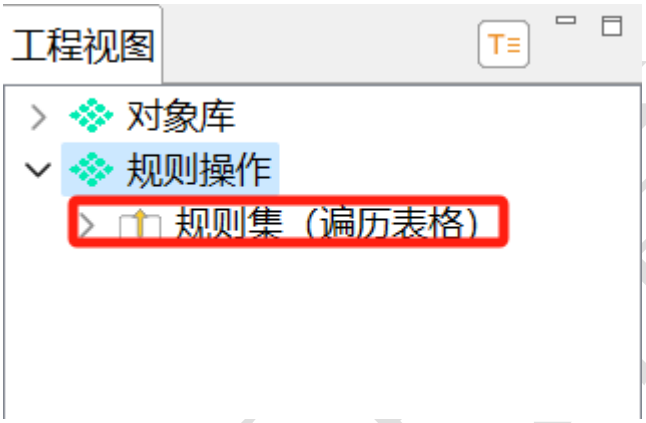
在规则配置过程中，当需要设置遍历表格或者编辑类型时，我们可以在规则的外层添加规则集。

3.1.1. 遍历表格

例如，在“员工信息”和“部门信息”中，要将这两张表的数据合并成一张表，这时我们需要添加规则集，设置双重表格遍历来分别获取两张表分数据。

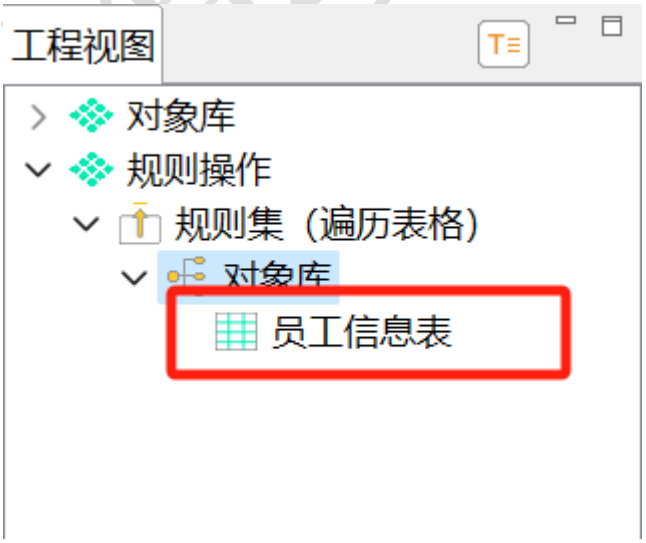
1．创建规则包

在“规则操作”工程下，右键添加规则包，并命名为“规则集（遍历表格）”：

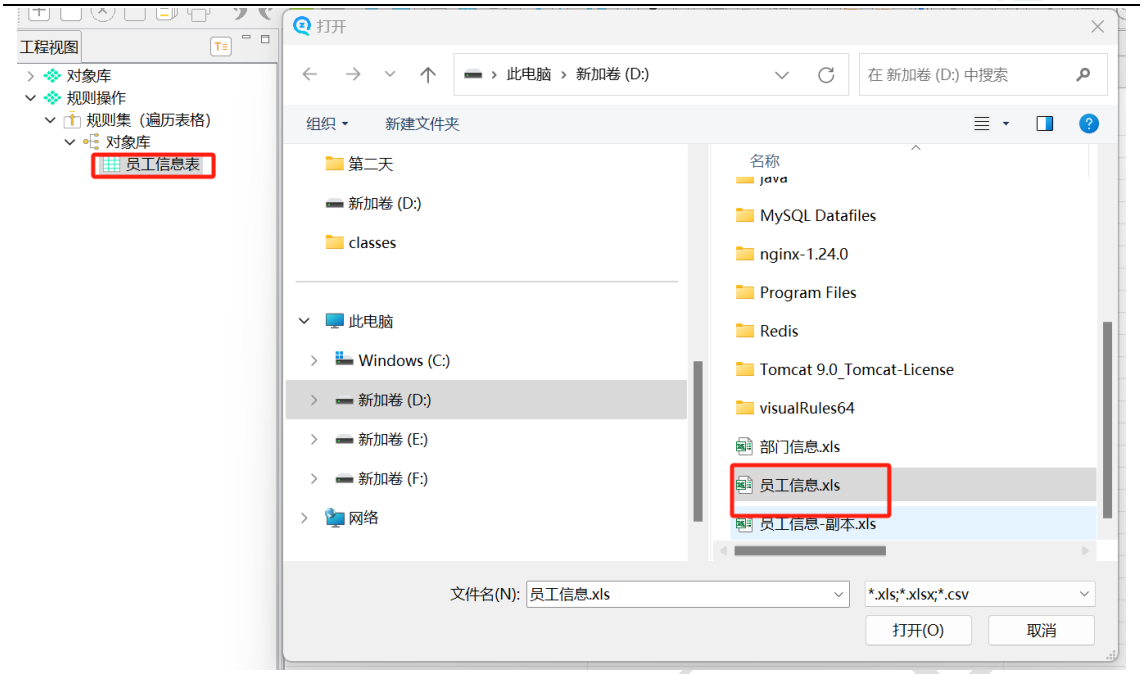


2．添加计算表格

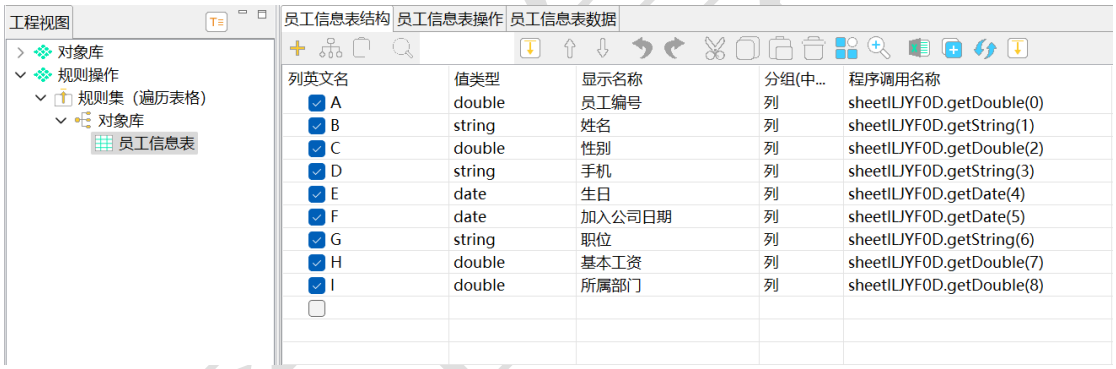
打开“规则集（遍历表格）”规则包，点击对象库，添加计算表格，并命名为“员工信息表”：



右键“员工信息表”，选择“从 Excel 中导入列信息”，在弹出框中选择“.xls”文件：

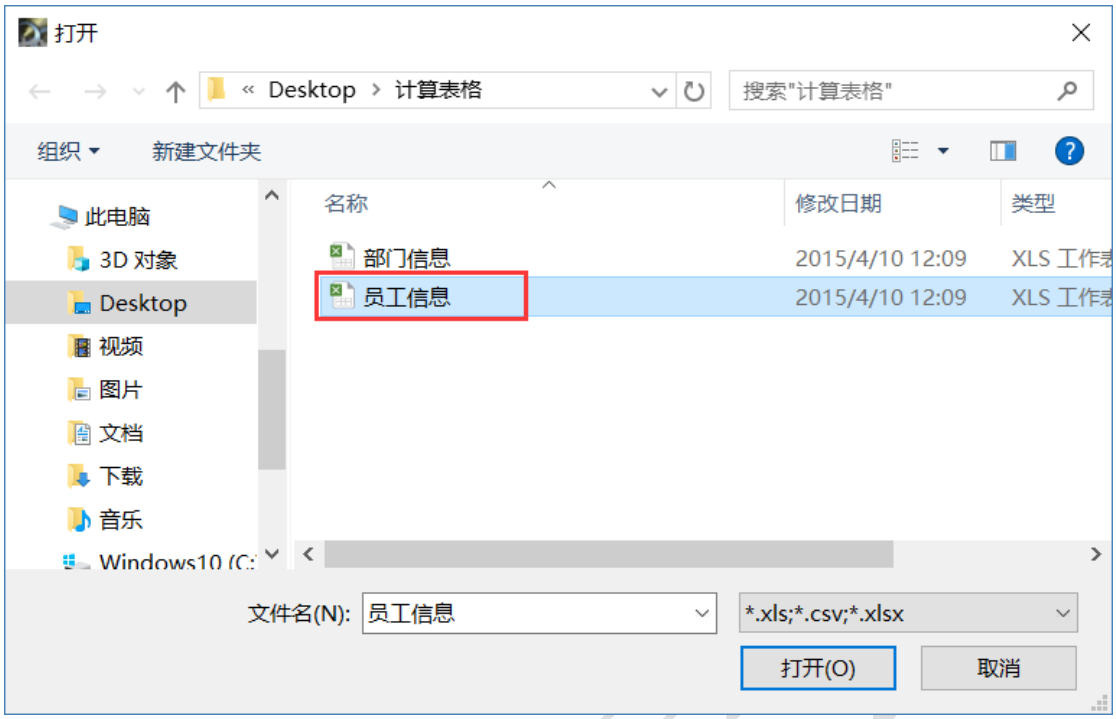


点击“员工信息表”查看列数据：

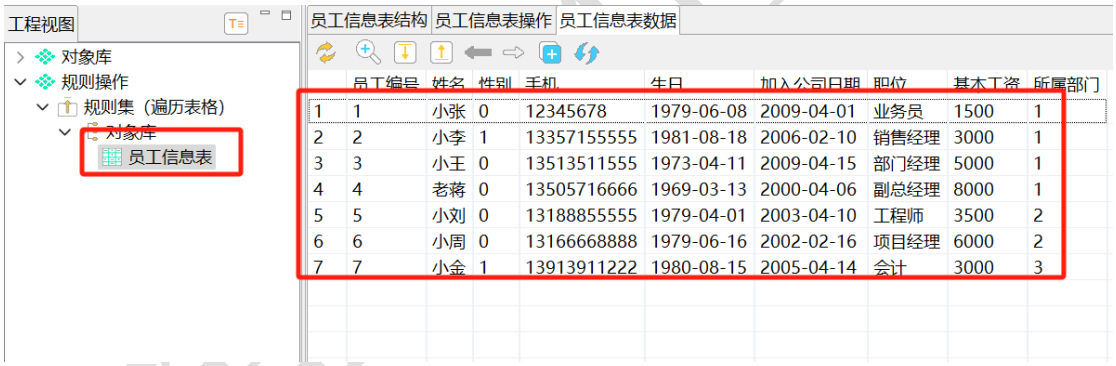


列英文名	值类型	显示名称	分组(中...	程序调用名称
A	double	员工编号	列	sheetILJYF0D.getDouble(0)
B	string	姓名	列	sheetILJYF0D.getString(1)
C	double	性别	列	sheetILJYF0D.getDouble(2)
D	string	手机	列	sheetILJYF0D.getString(3)
E	date	生日	列	sheetILJYF0D.getDate(4)
F	date	加入公司日期	列	sheetILJYF0D.getDate(5)
G	string	职位	列	sheetILJYF0D.getString(6)
H	double	基本工资	列	sheetILJYF0D.getDouble(7)
I	double	所属部门	列	sheetILJYF0D.getDouble(8)

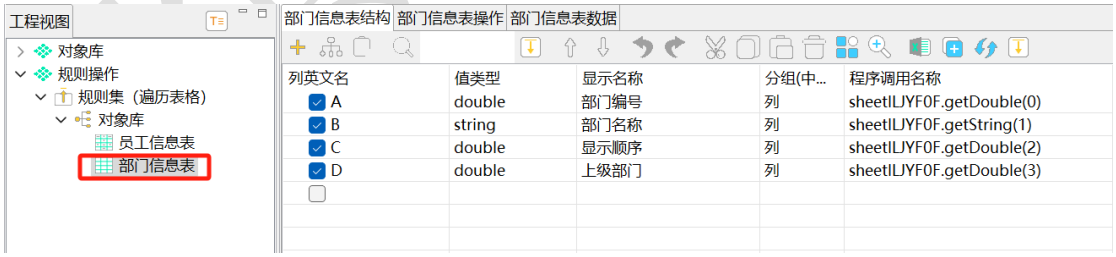
给员工信息表添加数据，右键点击“员工信息表”，选择“从 Excel 文件中导入数据”：

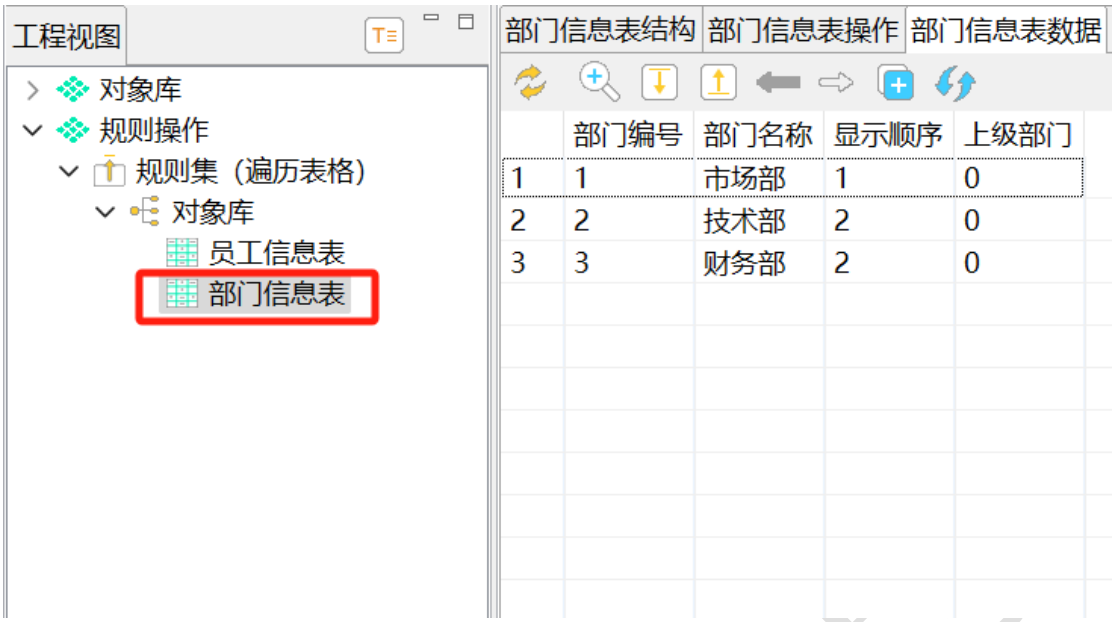


查看表数据：

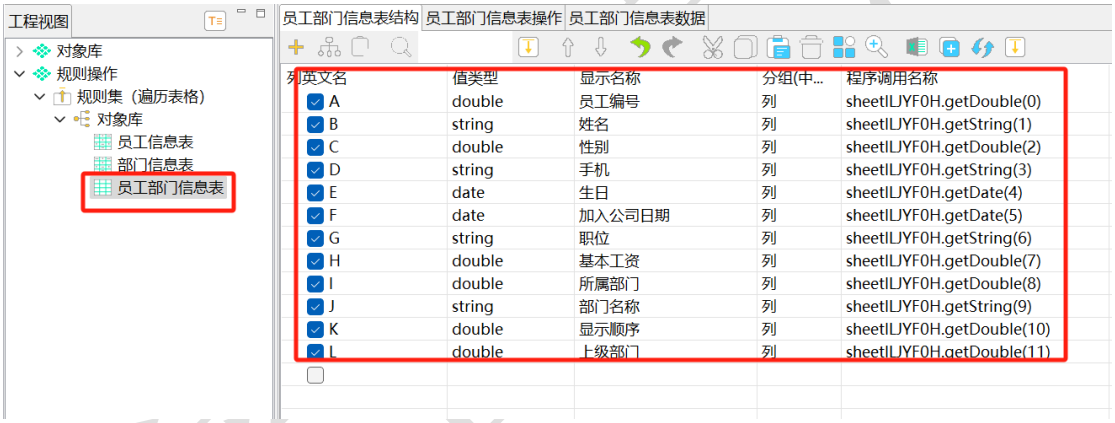


与上述方法相同，添加“部门信息表”：



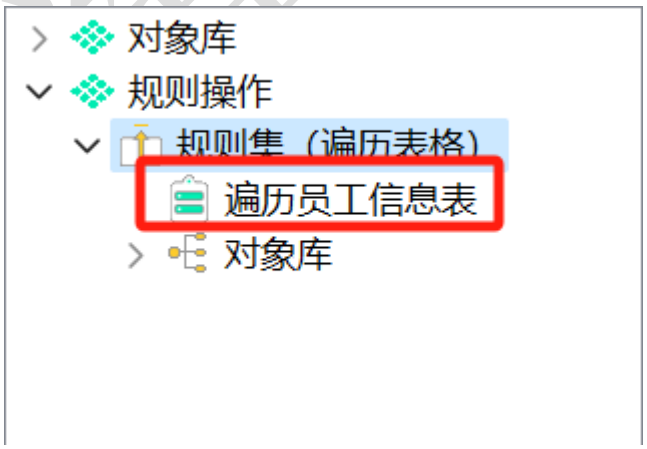


添加“员工部门信息表”，将“员工信息表”和“部门信息表”中的列添加到“员工部门信息”中，添加的时候去掉重复的列，如下图：



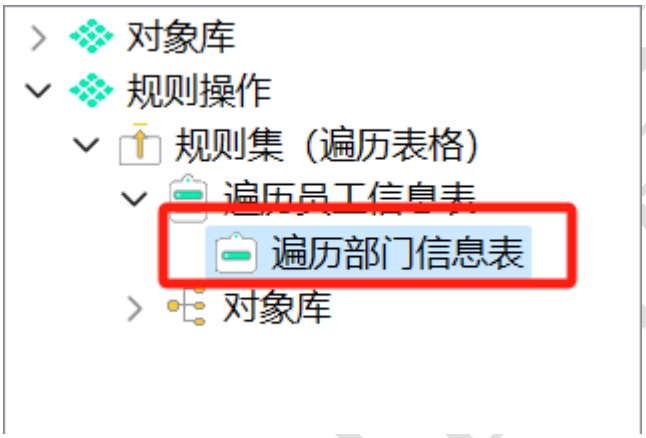
3．新建规则集

在“规则集（遍历表格）”下新建规则集，命名为“遍历员工信息表”：



打开“遍历员工信息表”规则集，在规则集中设置遍历表格，在属性窗口中，勾选遍历表格下方会新增一行，选择“员工信息表”：

在“遍历员工信息表”规则集下，新建“遍历部门信息表”规则集：

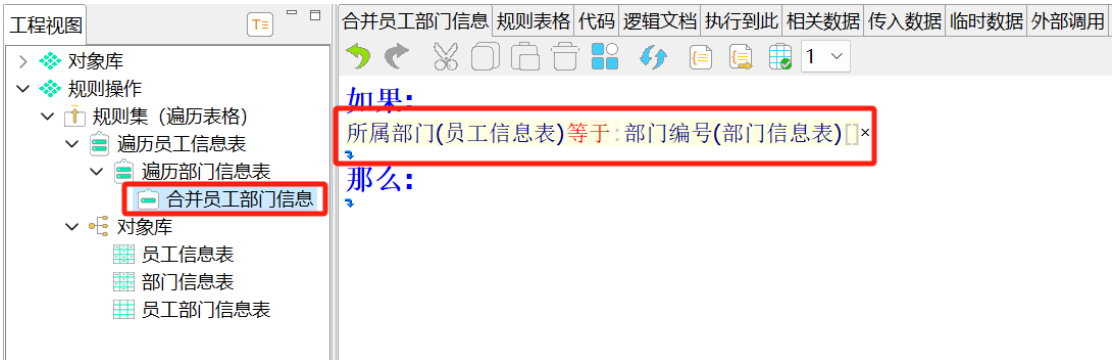


打开“遍历部门信息表”规则集，设置遍历表格，方法同上：

4．添加规则

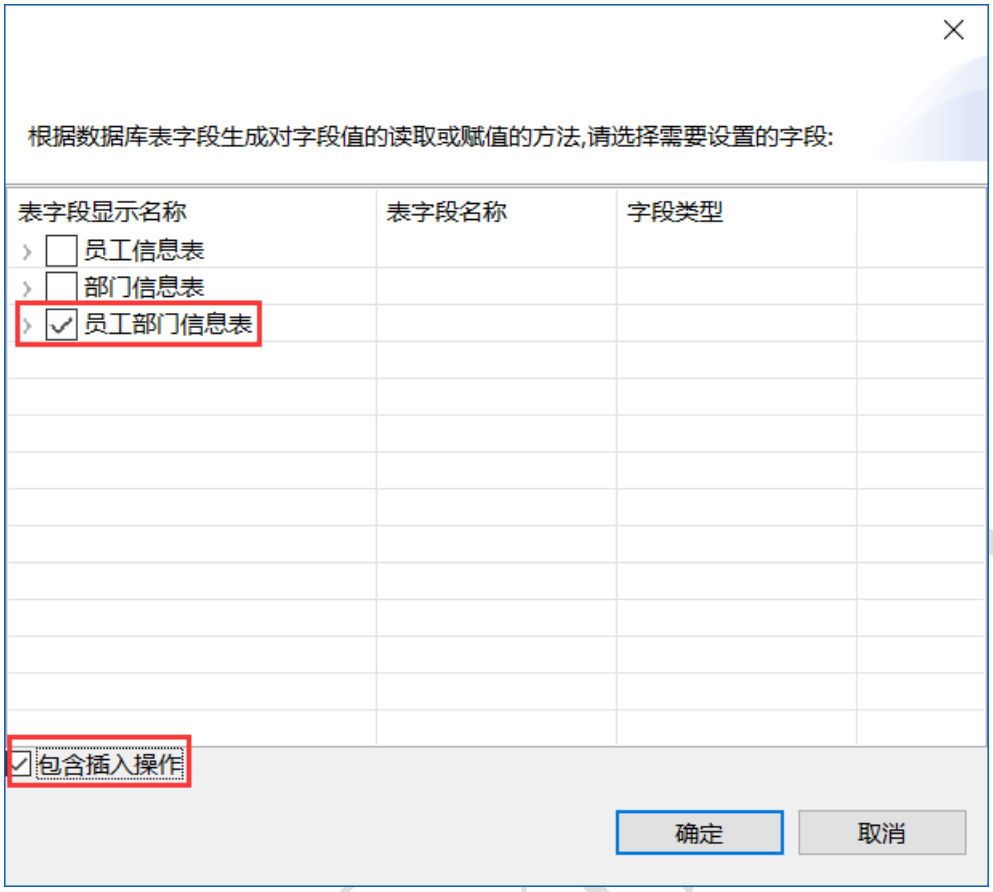
在“遍历部门信息表”规则集下新建规则，并命名为“合并员工部门信息”，在编辑窗

口中添加动作，通过“部门编号”设置条件：

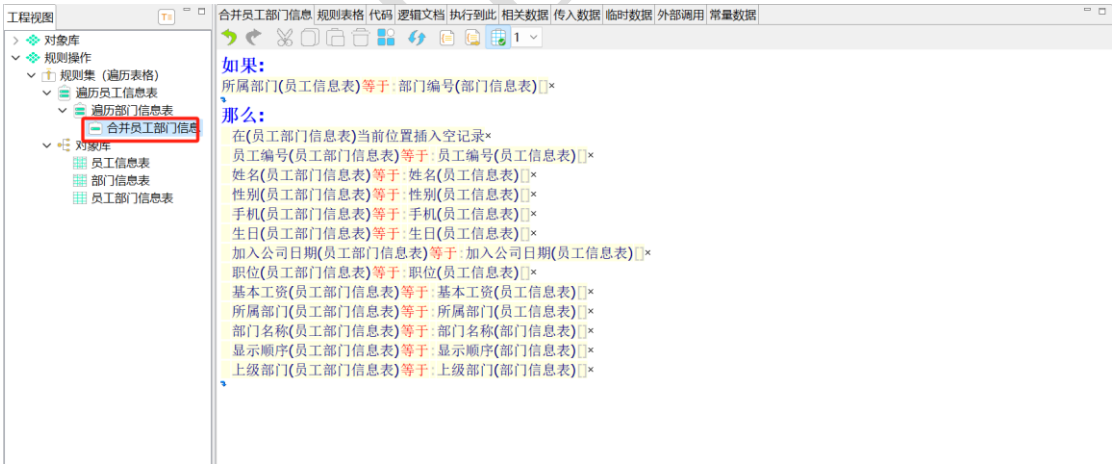


将“员工信息表”与“部门信息表”中的数据通过部门编号相等赋给“员工部门信息表”。右键点击“合并员工部门信息”，选择“内存表格操作向导”→对表格的列赋值向导：

在弹出框中勾选被赋值的表“员工部门信息表”并勾选“包含插入操作”：



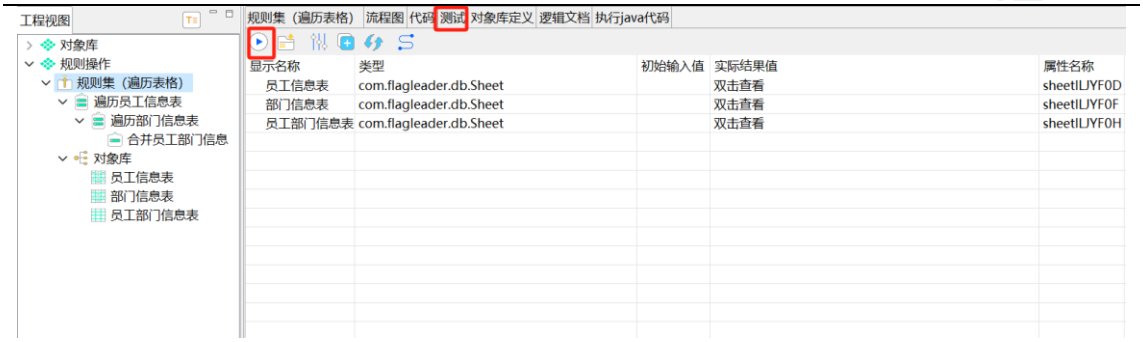
在遍历窗口的工具栏中，点击“ ”设置为“允许其他表格的列”：



保存并编译。

5．测试

点击“规则集（遍历表格）”，在测试选项卡中点击“ ”测试：



打开“员工部门信息表”，查看表数据：

3.1.2. 编辑类型

在规则集中，编辑类型共分为六种，分别是：没有条件、初始化和公共条件、循环运行、只有公共条件、逐个循环、同时满足多条件。默认类型为只有公共条件。

(1) 没有条件

当编辑类型设置为“没有条件”时，在编辑窗口中“进入条件”选项卡消失，此时规则集只做分类用，与规则组相同。

(2) 初始化和公共条件

当编辑类型为“初始化和公共条件”时，在编辑窗口中可以设置“初始化”变量和“进

入条件”。例如：

初始化：
i等于：0
进入条件：
i大于等于：0

代码为：

```
/* i等于"0" */
i=(int)0;
if (i>=0) {
}
```

当条件满足时，进入规则集中；条件不满足，则跳过此规则集。

(3) 循环运行

当编辑类型设置为“循环运行”时，如果条件满足则执行直至条件不成立时退出循环。例如：

初始化：
i等于：0
循环条件：
i小于：5
循环步骤：
i加上等于：1

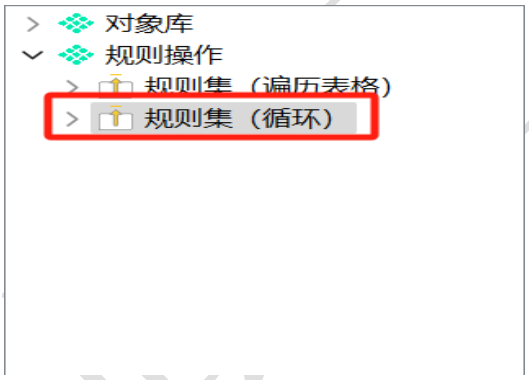
代码为：

```
/* i等于"0" */
i=(int)0;
while (i<5) {
   /* i加上等于"1" */
   i+=(int)1;
}
```

下面通过小案例来具体引入“循环运行”。

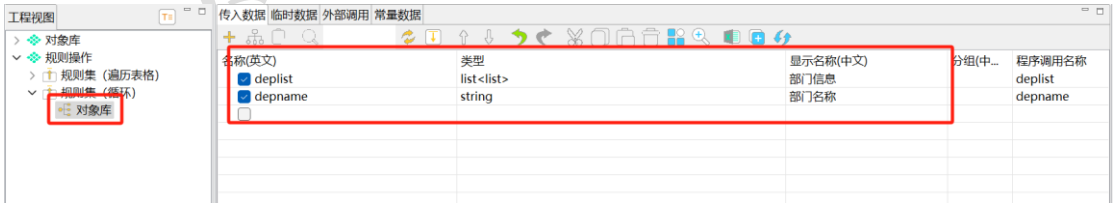
1．创建规则包

在“规则操作”工程下新建规则包，命名为“规则集（循环）”：

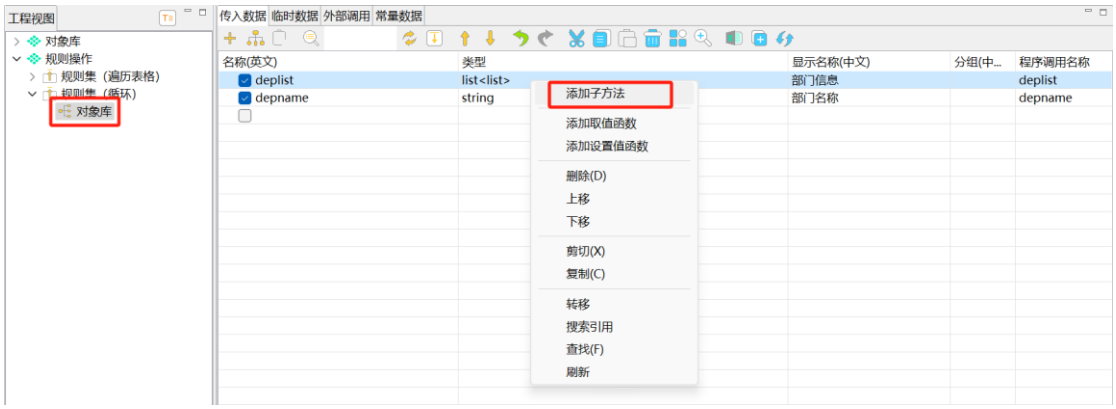


2．新增对象库变量

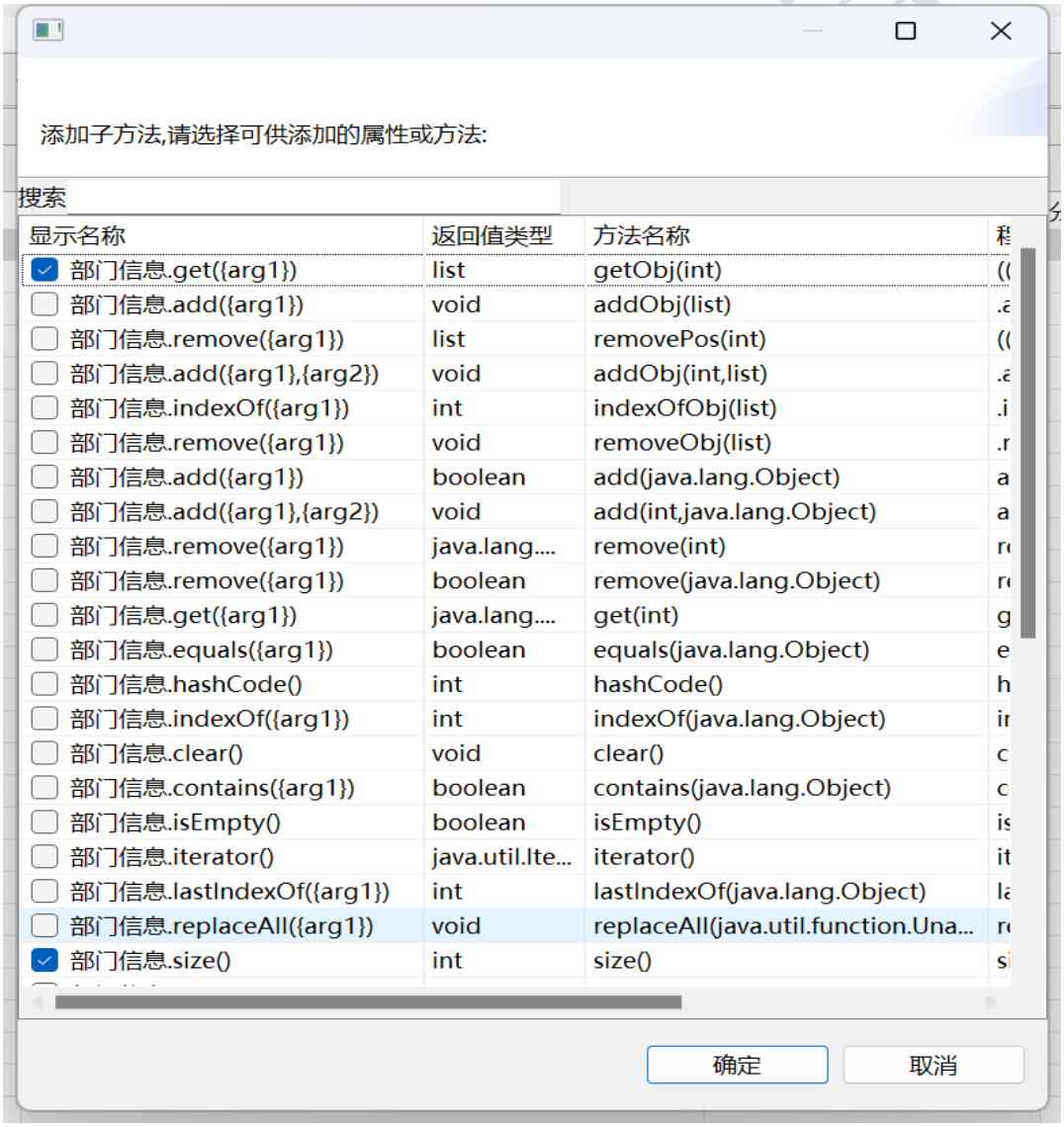
在对象库传入数据中添加“部门信息（deplist）”和“部门名称（depname）”变量：



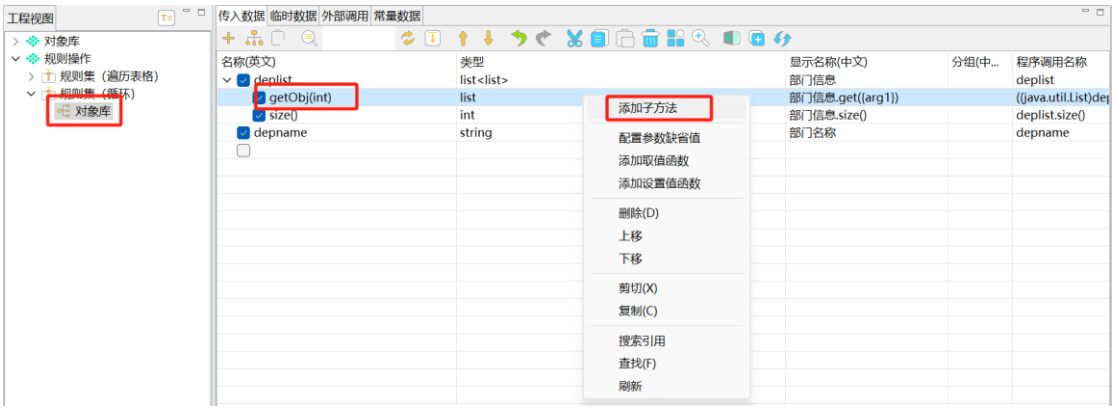
并给“部门信息”添加子方法：

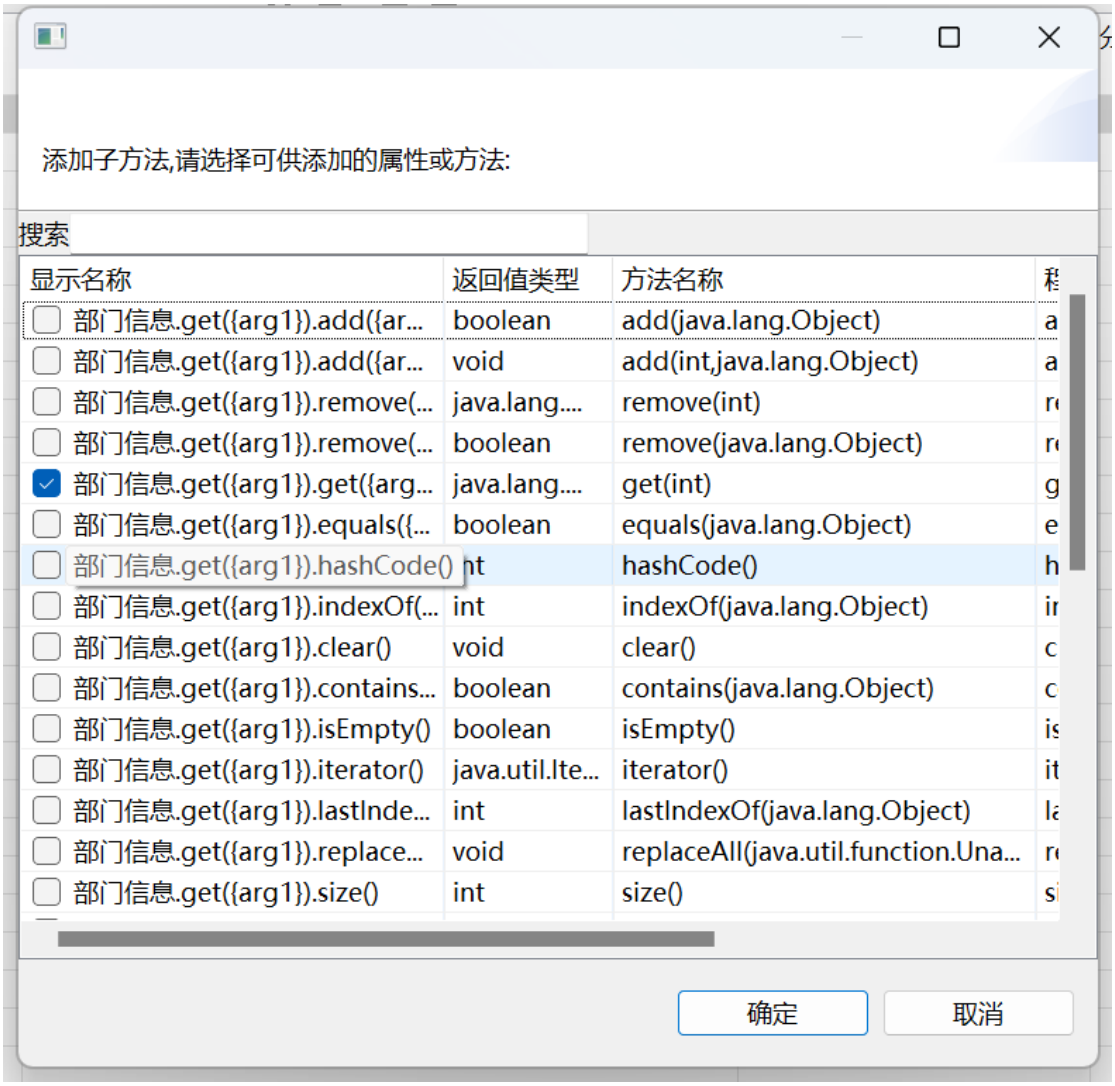


在弹出框中选择“getObj”和“size”方法，如下：

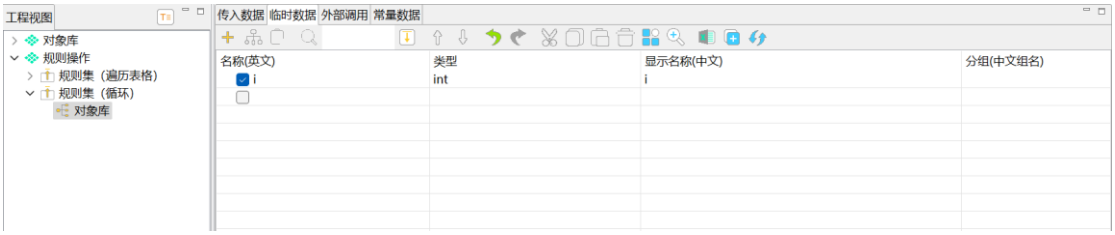


并给“getObj”方法添加子方法“get”：



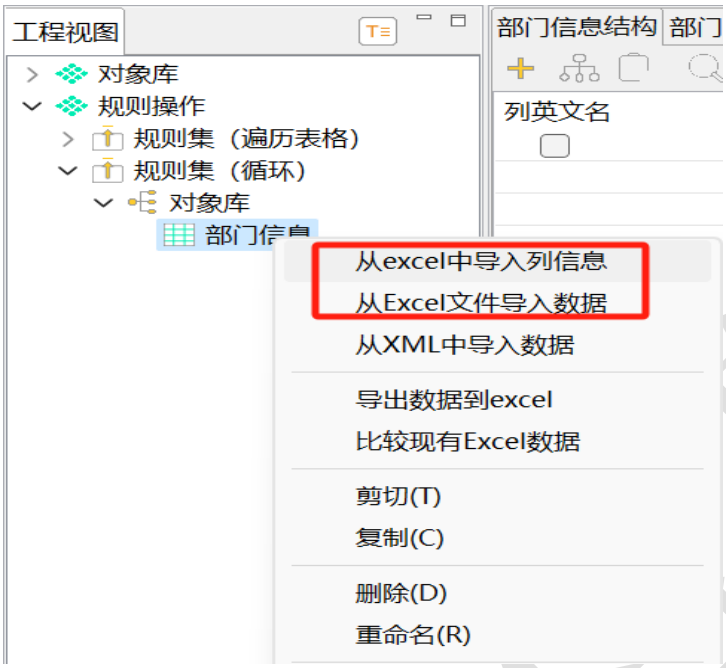


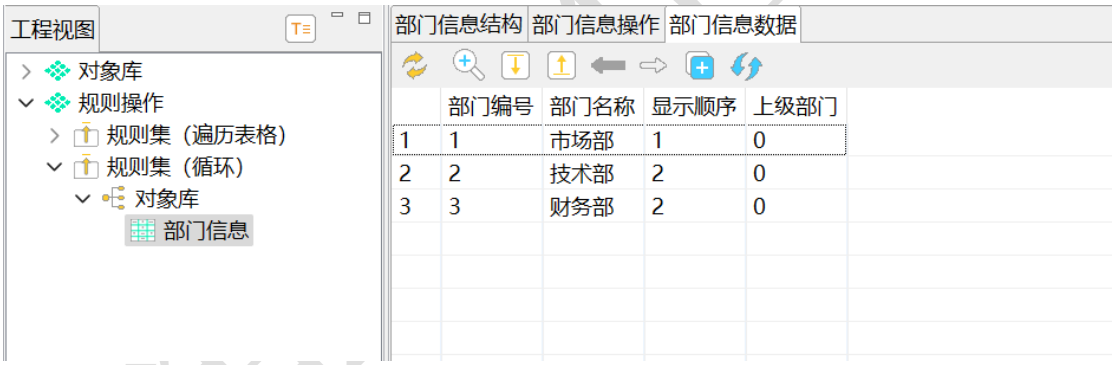
在“临时数据”中添加变量“i”：



3．添加计算表格

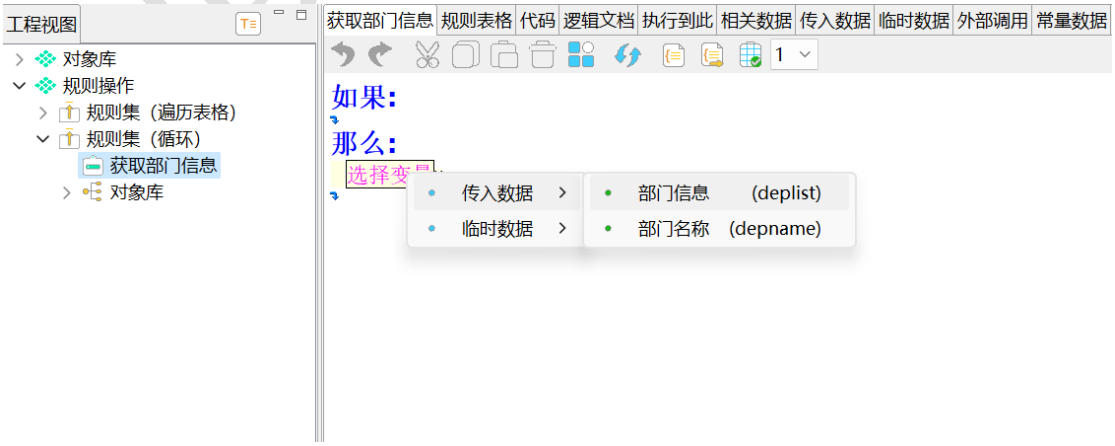
在对象库下新建计算表格，命名为“部门信息”，右键点击新建的“部门信息”，分别从Excel 中导入列和数据：





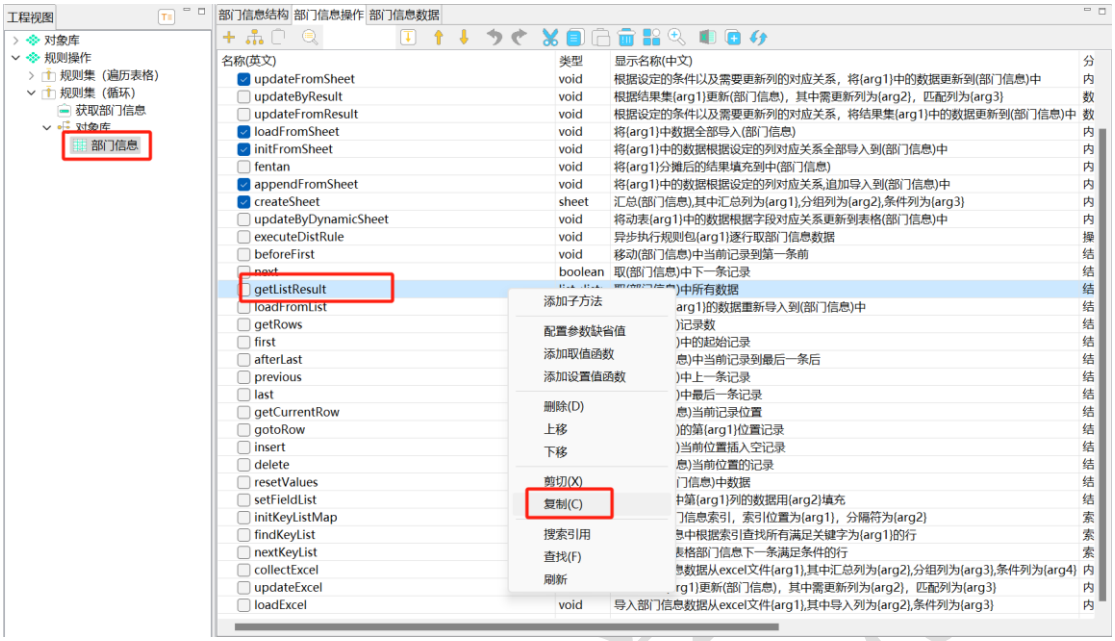
4．添加规则

在规则包 下新建规则，命名为“获取部门信息”，并添加动作：

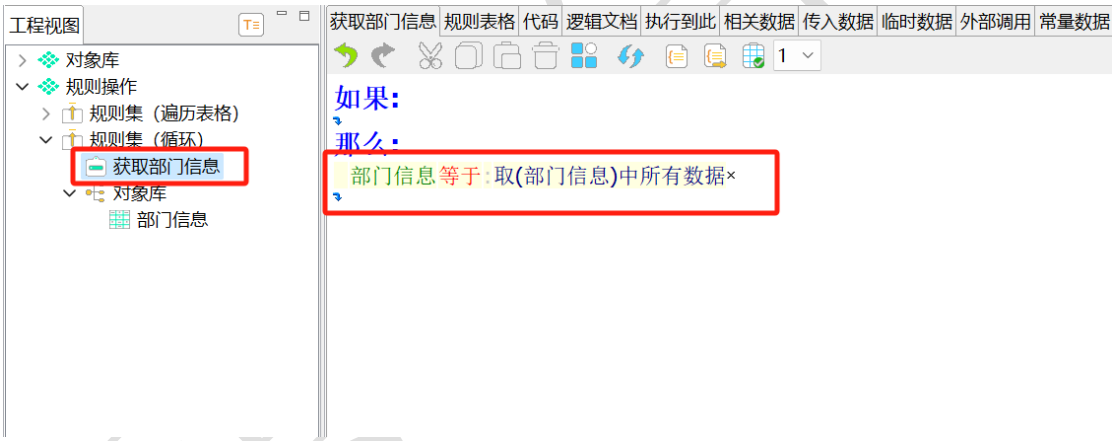


将“部门信息”表中的数据全部赋给传入数据“部门信息”。在“部门信息”表中，点

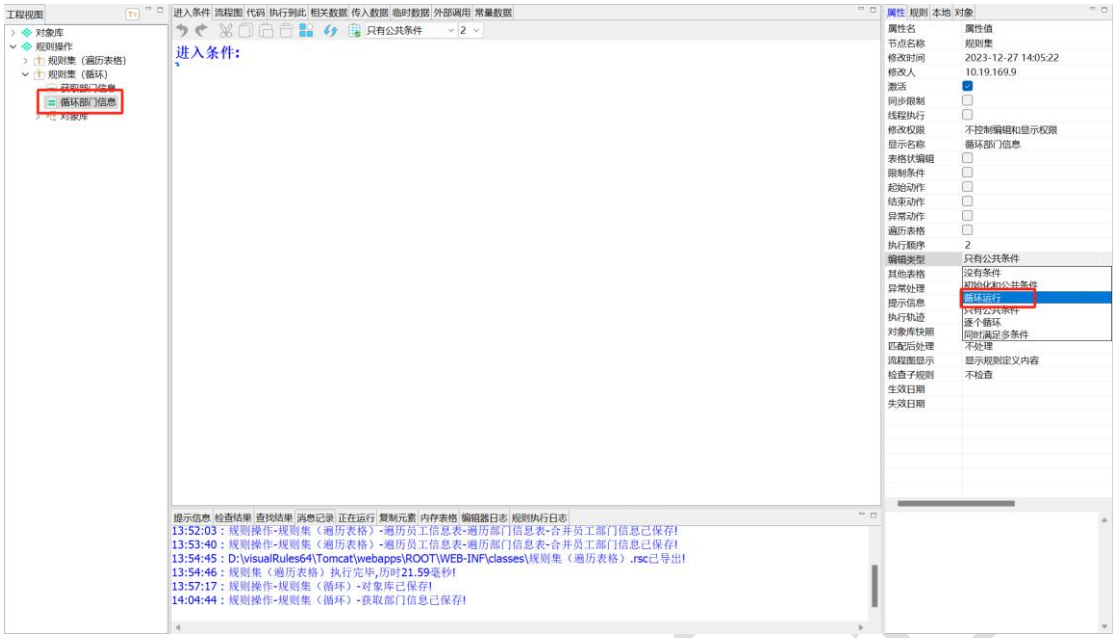
击操作选项卡，选择“取（部门信息）中的所有数据”复制：



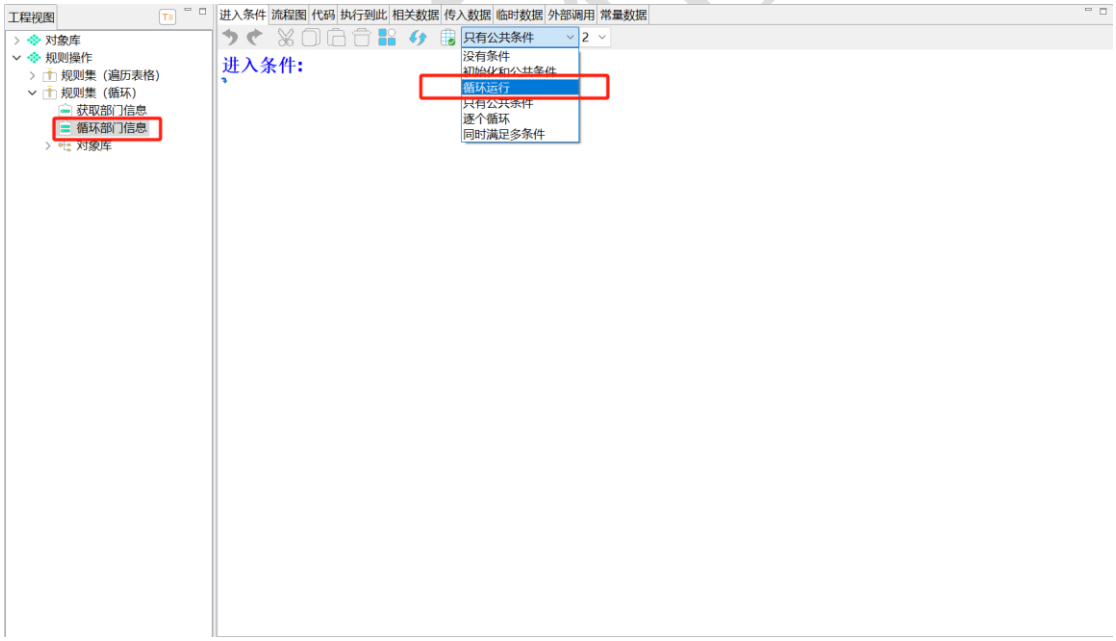
在“获取部门信息”规则中通过粘贴给“部门信息”变量集合赋值：



新建规则集，命名为“循环部门信息”，在右侧属性窗口中，将编辑类型的属性值由默认的“只有公共条件”改为“循环运行”：

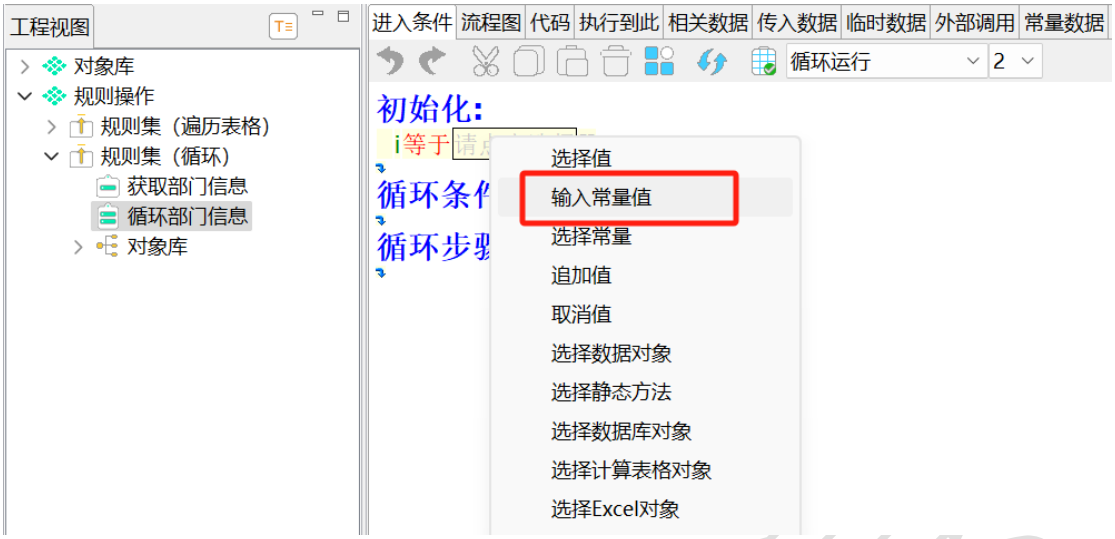


或者在编辑窗口的“进入条件”选项卡中，在工具栏中通过点击“ 只有公共条件 ”，在下拉列表中进行更换：

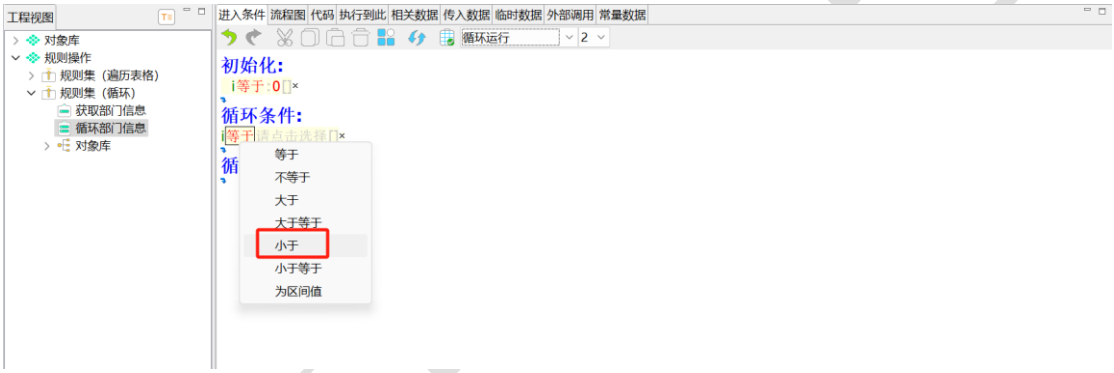


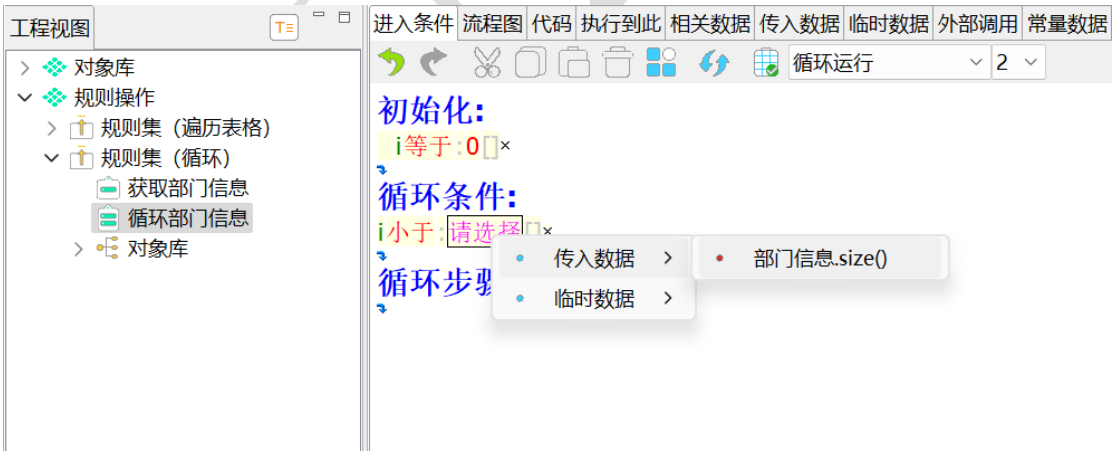
设置“初始化”、“循环条件”、“循环步骤”。

初始化临时数据 i 的值为 0：

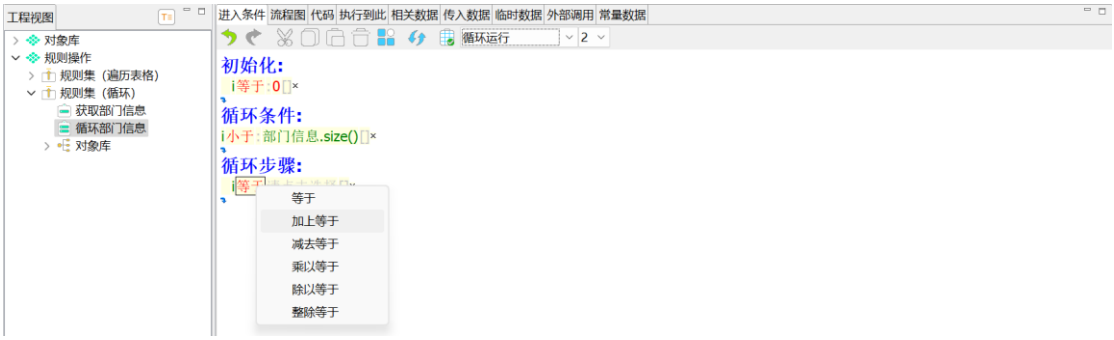


循环条件为 i 小于部门信息.size：

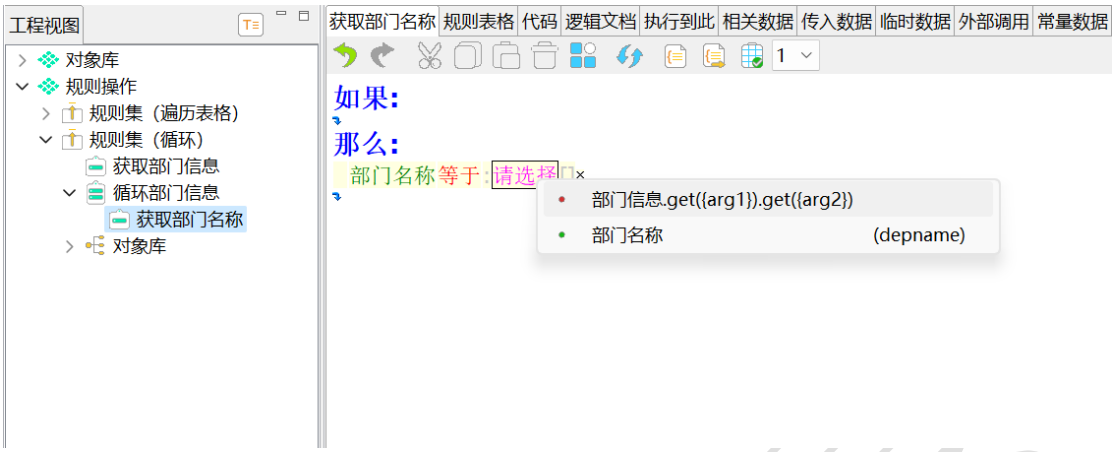




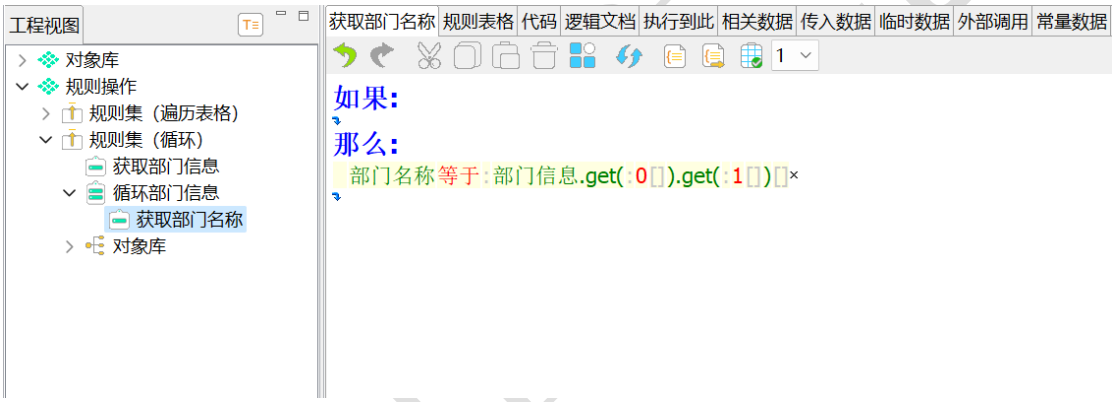
循环步骤为 i 加上等于 1：



在“循环部门信息”规则集下新建规则，命名为“获取部门名称”，并添加动作：



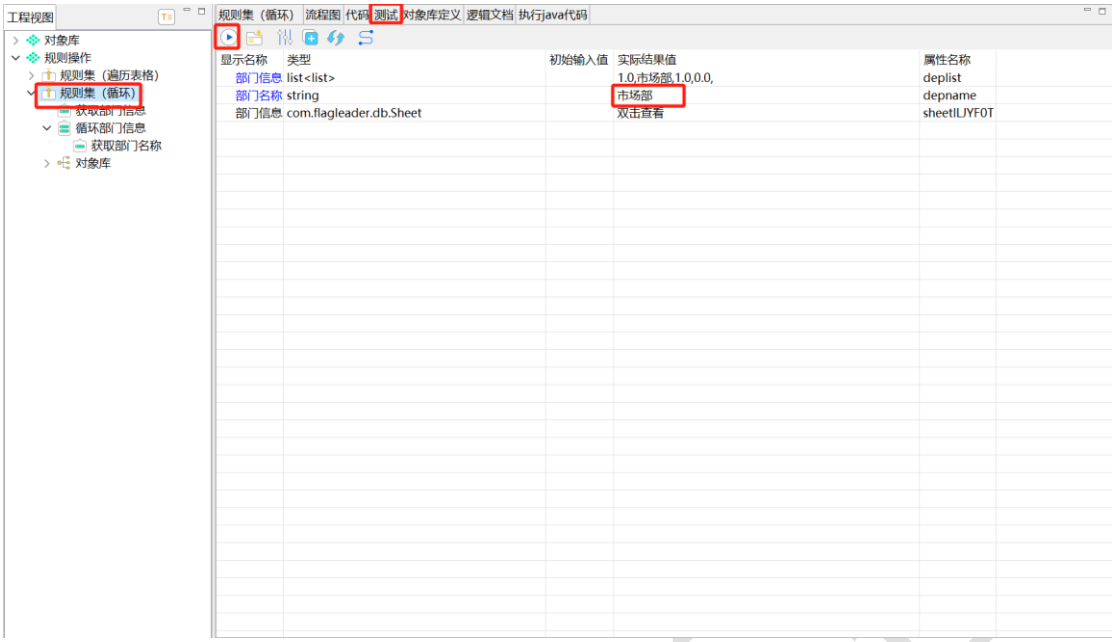
在“部门信息.get({arg1}).get({arg2})”中，第一个 get 是获取行，第二个 get 是获取列，这里 arg1 输入“0”，arg2 输入“1”为例：



保存并编译。

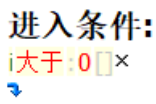
5．测试

点击“规则集（循环）”规则包，在测试界面中测试如下：



(4) 只有公共条件

当编辑类型为“只有公共条件”时，可以设置“进入条件”：



代码为：

```
if (i>0) {
}
```

满足条件则进入，不满足则跳过。

(5) 逐个循环

当编辑类型为“逐个循环”时，已生成循环条件，只需给其赋值，如下：

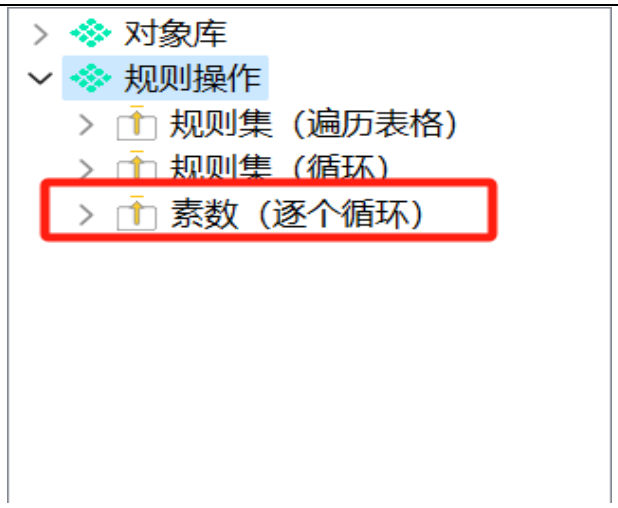
代码为：

```
for (i = 0 ; i <= 5 ; i++ ) {
}
```

这里通过在规则引擎中“取 1-100 之间的素数”来引用该循环。

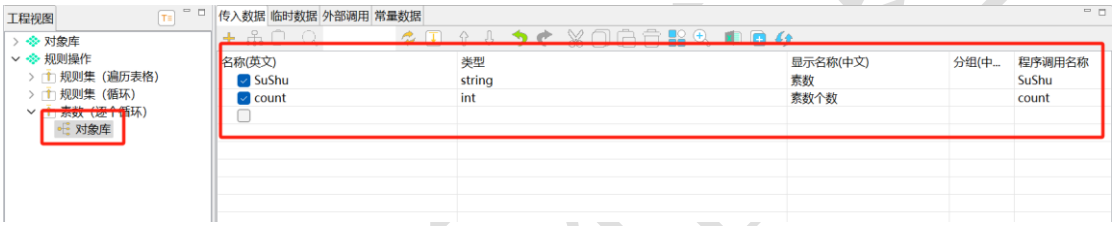
1．创建规则包

在“规则操作”工程下，新建“素数（逐个循环）”规则包：

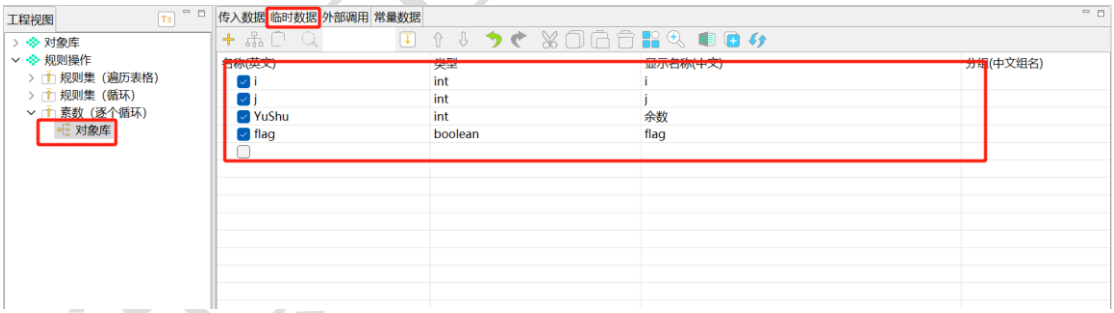


2．定义变量

在对象库传入数据中定义如下变量：

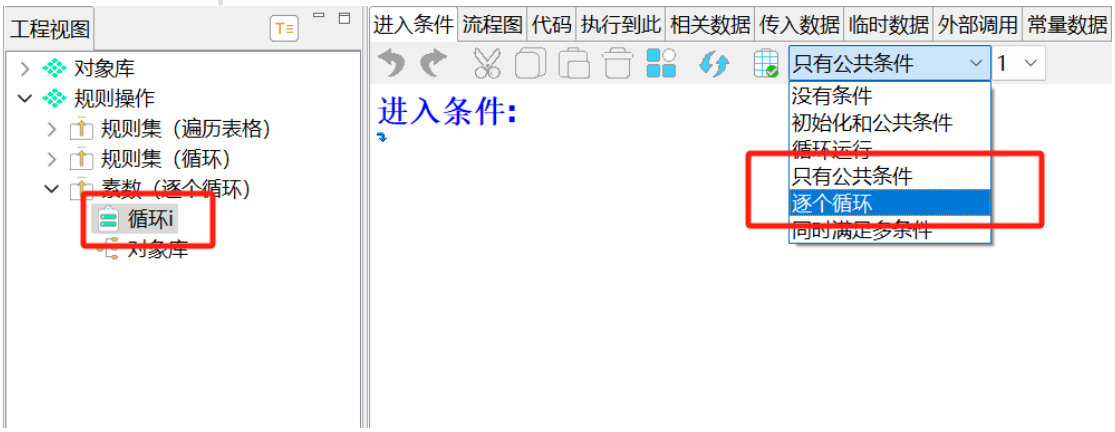


在临时数据中定义以下变量,其中 flag 的类型为 boolean:

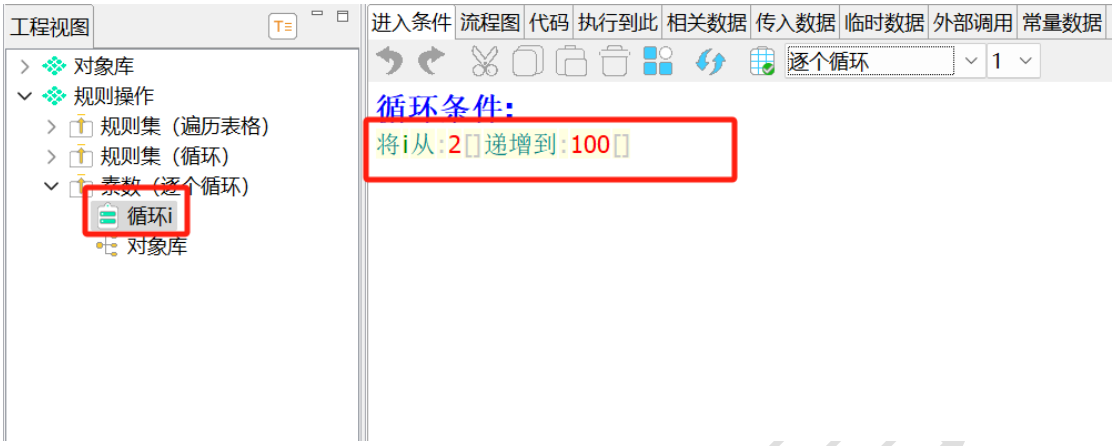


3．新建规则和规则集

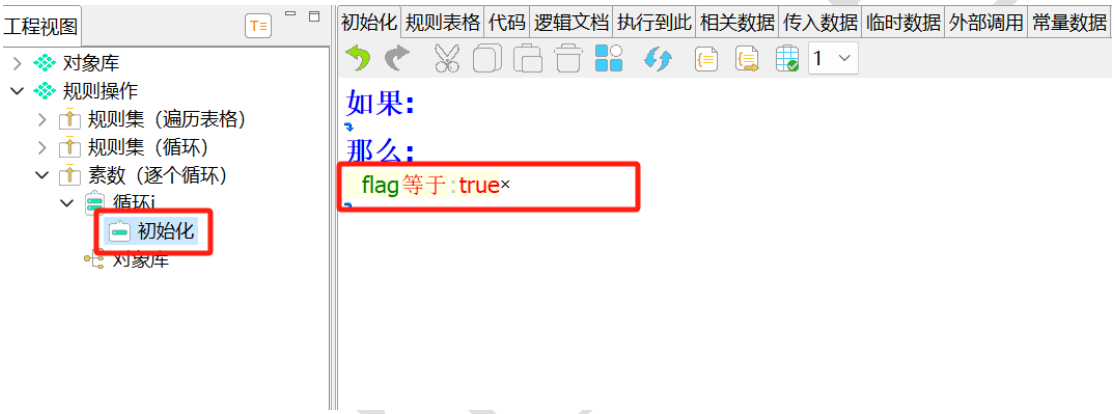
在规则包下新建规则集，命名为“循环 i”，并设置编辑类型为“逐个循环”：



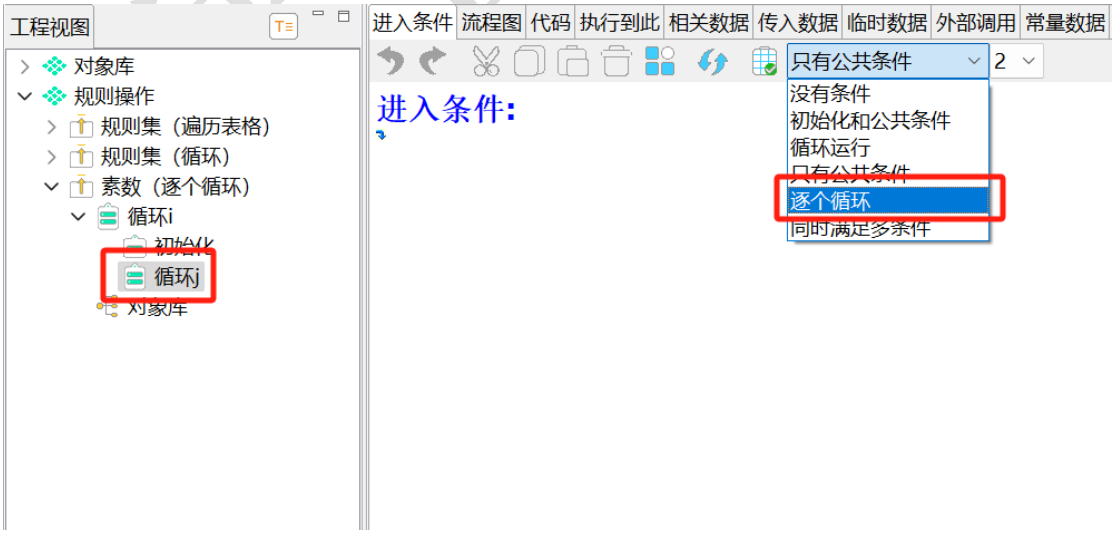
选择临时数据中的变量 i,并从 2 递增到 100：



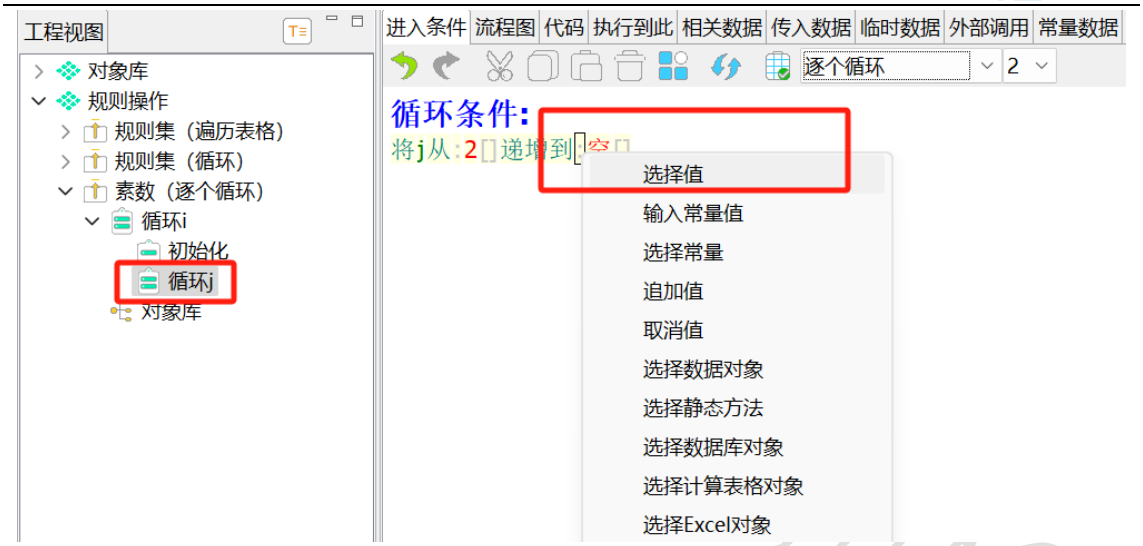
在规则集“循环 i”下添加规则“初始化”，初始化“flag”的值为 true：



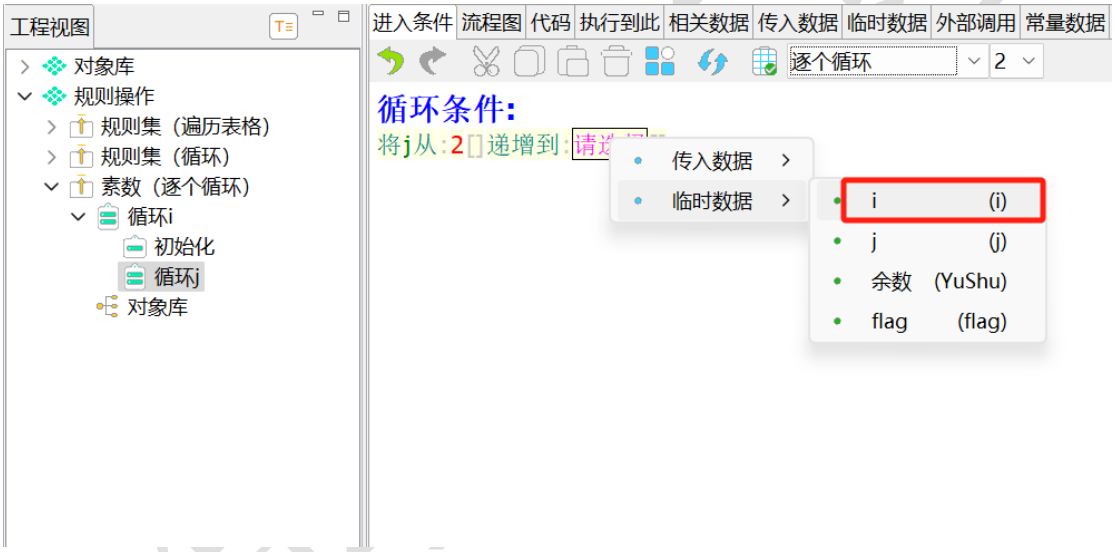
再在规则集“循环 i”下添加规则集，命名为“循环 j”，并设置编辑类型为“逐个循环”：



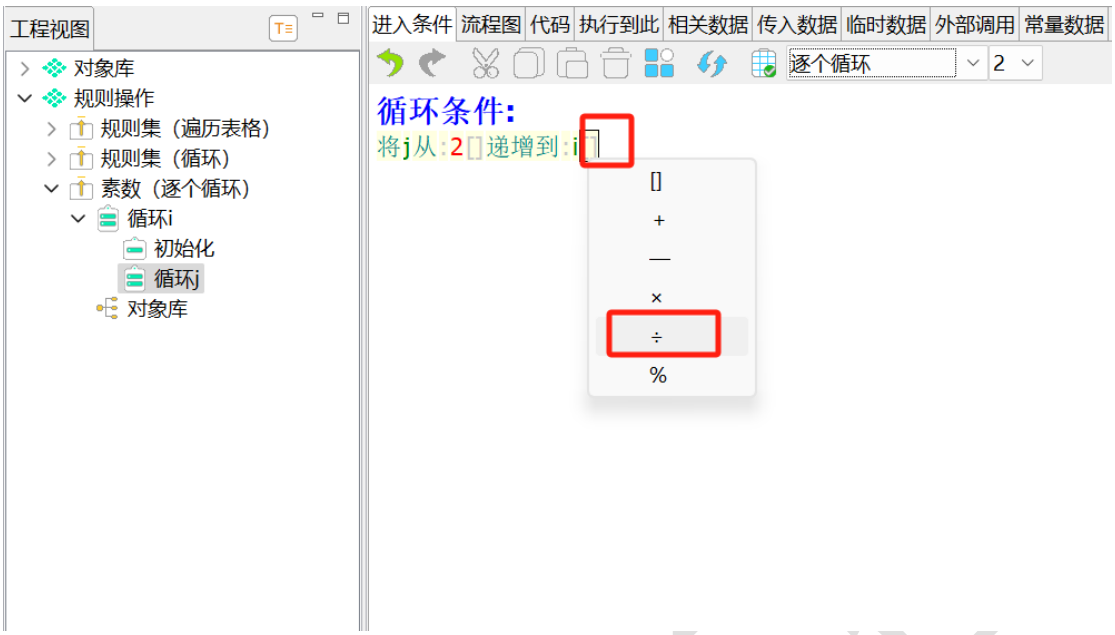
选择临时数据中的变量 j，并从 2 递增到 i/2:

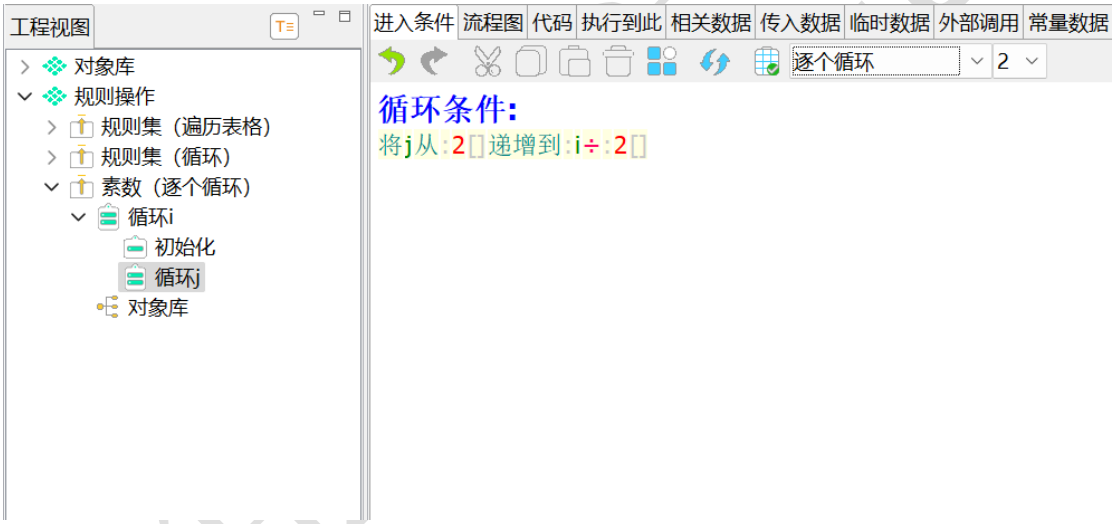


选择选择值→临时数据→i:

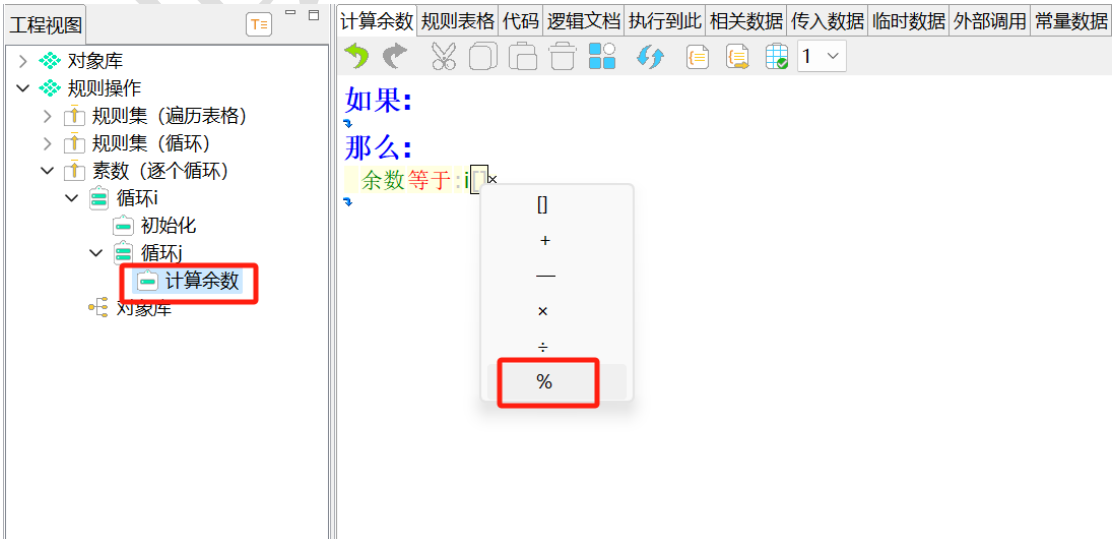


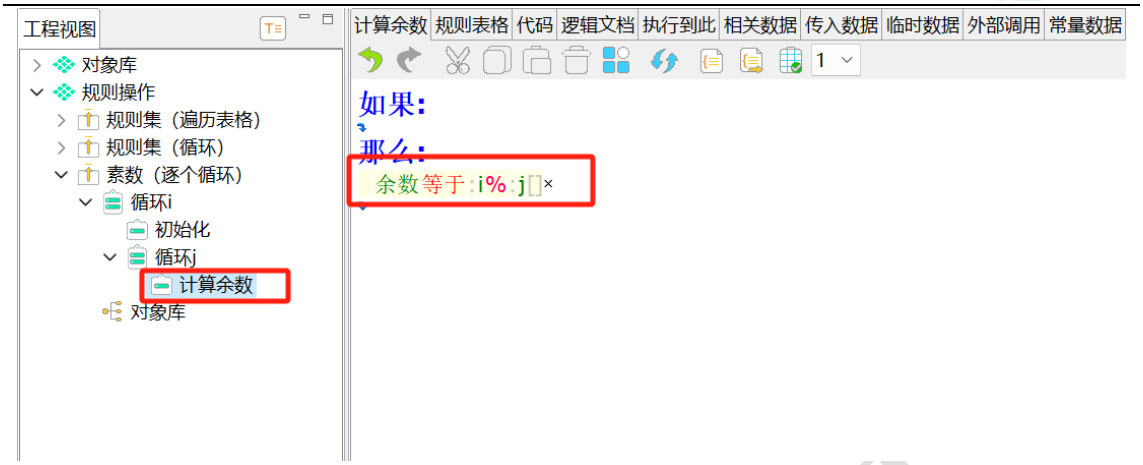
点击“[]”选择数值间操作符：



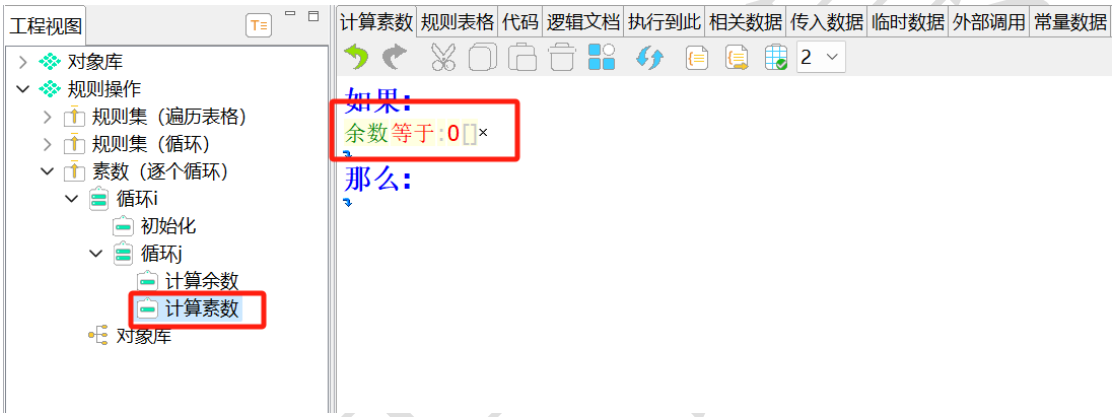


在“循环 j”规则集中添加规则“计算余数”，并添加动作：

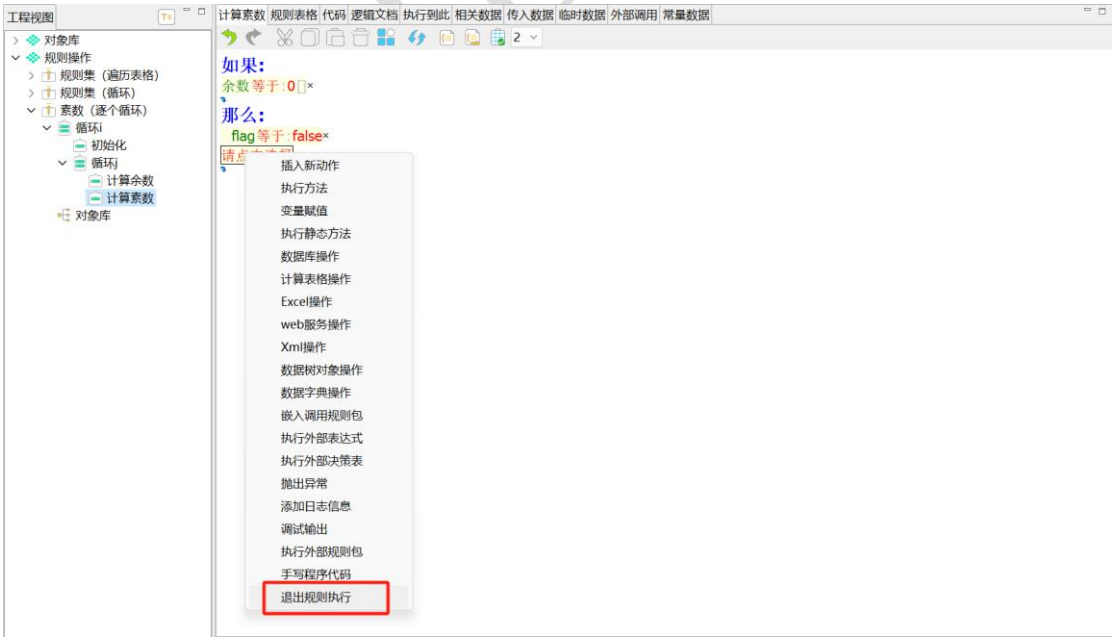




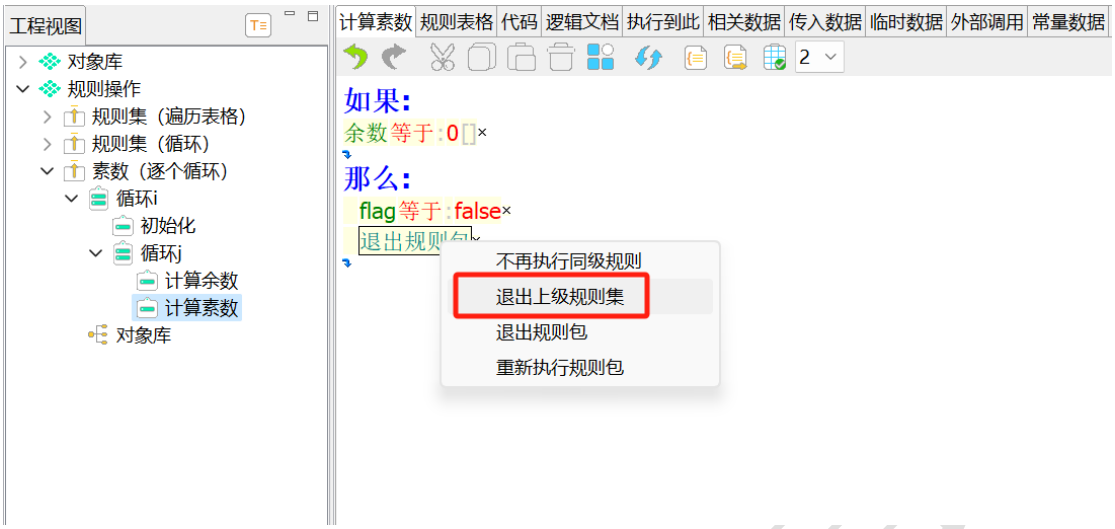
在规则集“循环 j”下添加规则“计算素数”，在“如果”中添加条件：



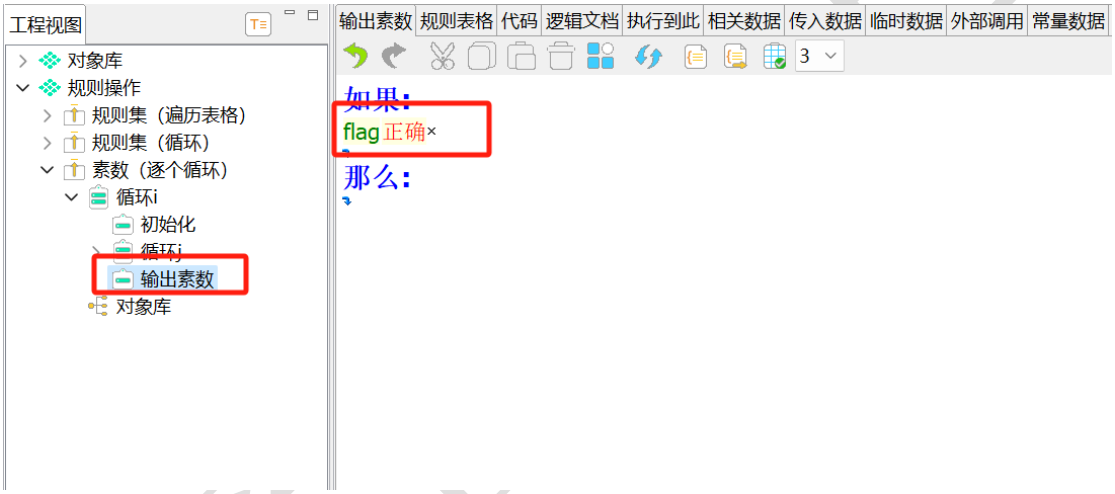
添加动作，将“flag”的值更改为“false”并跳出循环：



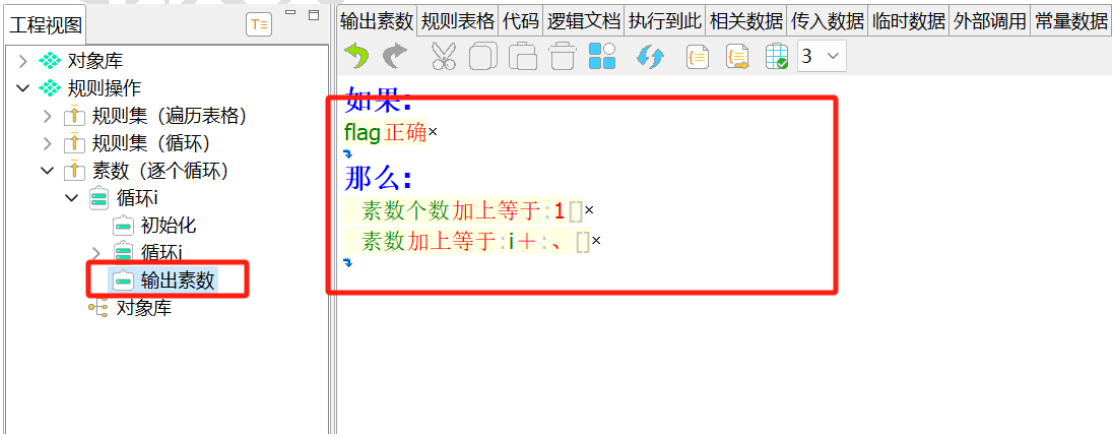
点击“退出规则包”，选择“退出上级规则集”：



在规则集“循环 i”下添加规则包，命名为“输出素数”，在“如果”中添加条件：



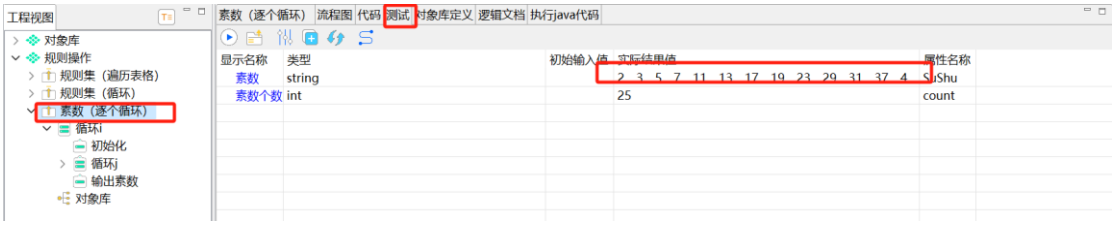
添加动作，得到素数的个数，和所有素数的值：



保存并编译。

4．测试

点击“素数（逐个循环）”规则包，选择“测试”选项卡，点击进行测试：



以下是该规则程序的 Java 代码：

```
for (i = 2 ; i <= 100 ; i++ ) {
    /* flag 等于"true" */
     flag=true;
     for (j = 2 ; j <= i/2 ; j++       ) {
        /* 余数等于 i%j */
         YuShu=(int)i%j;
/*  余数等于"0"    */
         if ( YuShu==0 ) {
            /* flag 等于"false" */
              flag=false;
            /* 退出上级规则集 */
              if ( true ) return M_RETURN_BREAK ;
         }
     }
/*  flag 正确    */
     if ( flag ) {
        /* 素数的个数加上等于"1" */
         count+=(int)1;
        /* 素数加上等于 i+"、" */
         SuShu+=StringUtil.add(String.valueOf(i),"、");
     }
}
```

(6) 同时满足多条件

当编辑类型设置为“同时满足多条件”时，在此规则集中设置的众多条件中只需满足其中的一个条件即可通过。

3.2. 关联决策表

1. 业务需求

在计算个人所得税时，由于“税率”和“速算扣除数”都是根据“全月所得税额”的变化而变化的。若我们采用常规的方式来处理，则需写很多”if”、”else if”语句；而采用关联决策表的方式，把“全月所得税额”作为条件，把“税率”和“速算扣除数”作为结果，则可以省去许多冗余的逻辑，使程序员开发更加方便。现有的个人所得税的“税率”计算方法如下图所示：

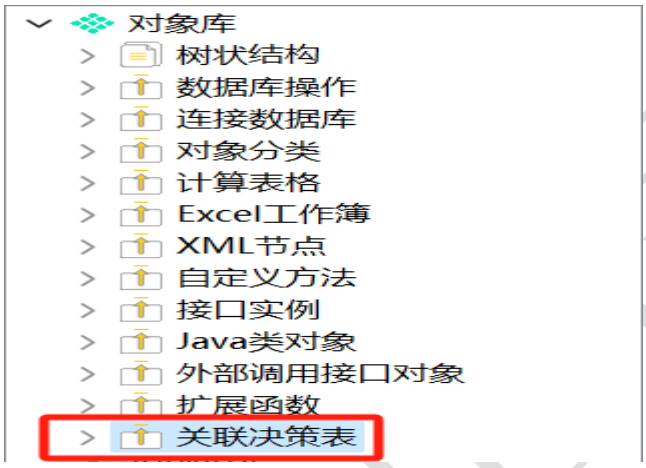
全月应纳税所得额	税率	速算扣除数（元）
全月应纳税所得额不超过1500元	3%	0
全月应纳税所得额超过1500元至4500元	10%	105
全月应纳税所得额超过4500元至9000元	20%	555
全月应纳税所得额超过9000元至35000元	25%	1005
全月应纳税所得额超过35000元至55000元	30%	2755
全月应纳税所得额超过55000元至80000元	35%	5505
全月应纳税所得额超过80000元	45%	13505

这个例子中就是根据已知“当月基本工资”，根据上述的税率方式计算“当月实际工资”。

2. 规则实现

(1) 创建规则包

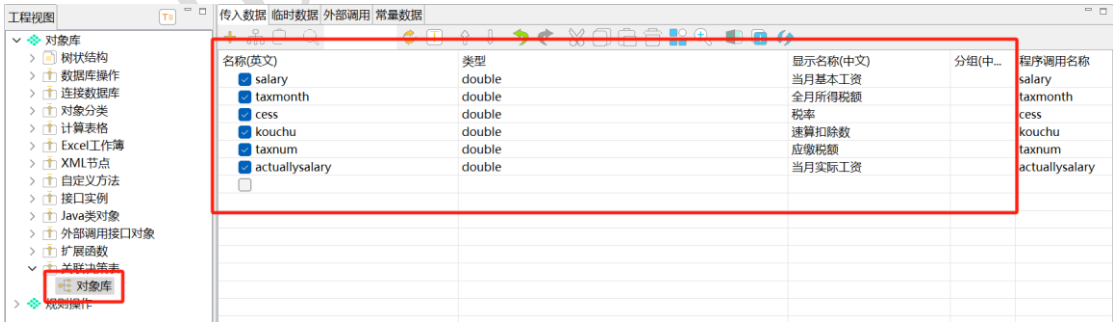
右键名为“对象库”的规则工程，点击“新建规则包”，创建一个名为“关联决策表”的规则包，如下图所示：



(2) 定义变量

我们需要在该规则包的对象库中定义六个变量：当月基本工资（salary），全月所得税额（taxmonth），税率（cess），速算扣除数（kouchu），应缴税额（taxnum），当月实际工资（actuallysalary）。

选择“关联决策表”规则包，点击“对象库”在“传入数据”中依次添加这六个变量，如下图所示：



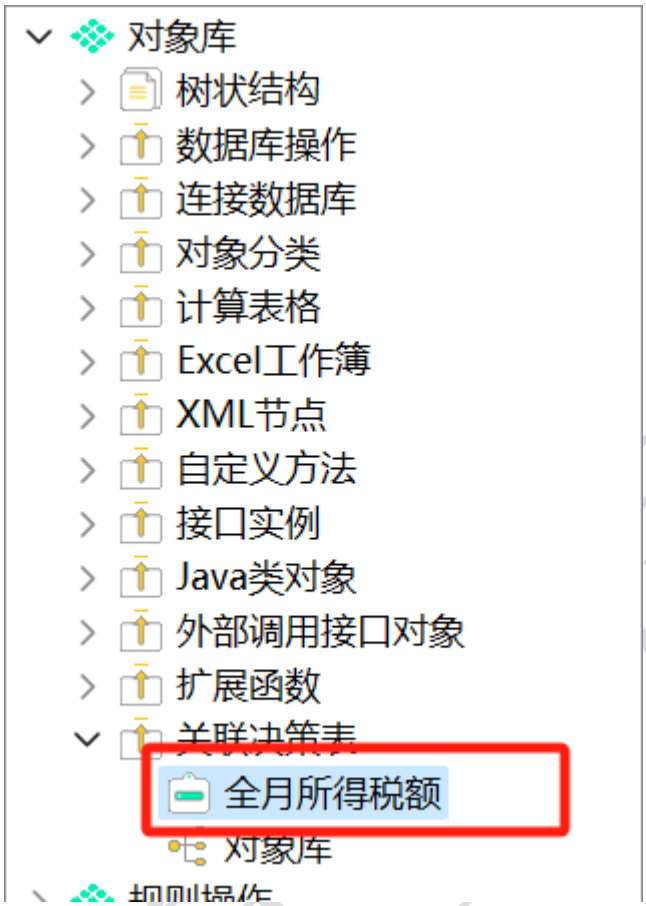
(3) 逻辑实现

首先，在计算“当月实际工资”时，必要得到 “全月所得税额”的值（全月所得税额=当月基本工资-3500），再根据“全月所得税额”计算“税率”和“速算扣除数”，最后再根

据“当月基本工资”和“应缴税额”得出“当月实际工资”（应缴税额=全月所得税额*（税率/100）—速算扣除数，当月实际工资=当月基本工资—应缴税额）。逻辑理好后，我们根据这个逻辑去创建规则及关联决策表。

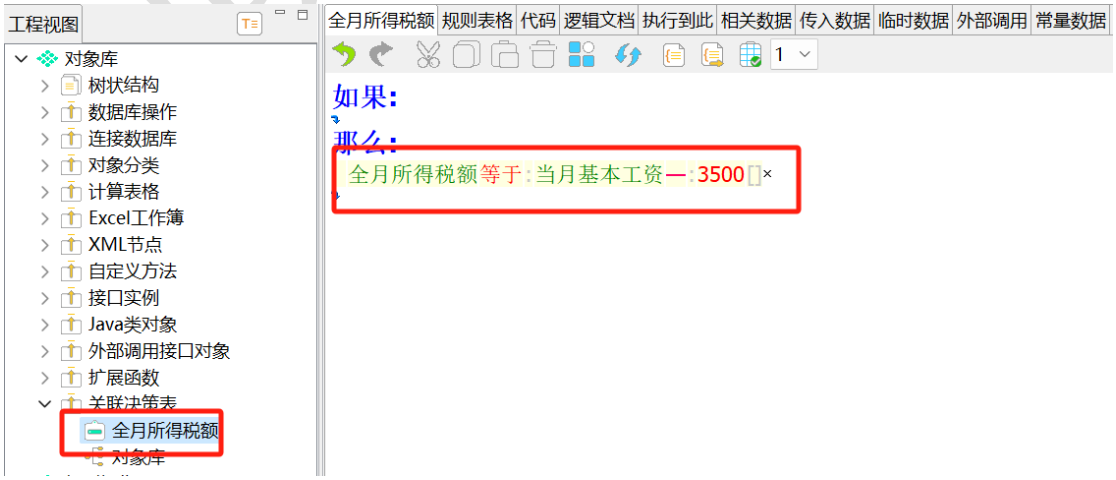
(4) 创建“全月所得税额”规则

右键规则包，创建名为“全月所得税额”的规则，如下图所示：



[注：在规则引擎中规则是最小的单位。]

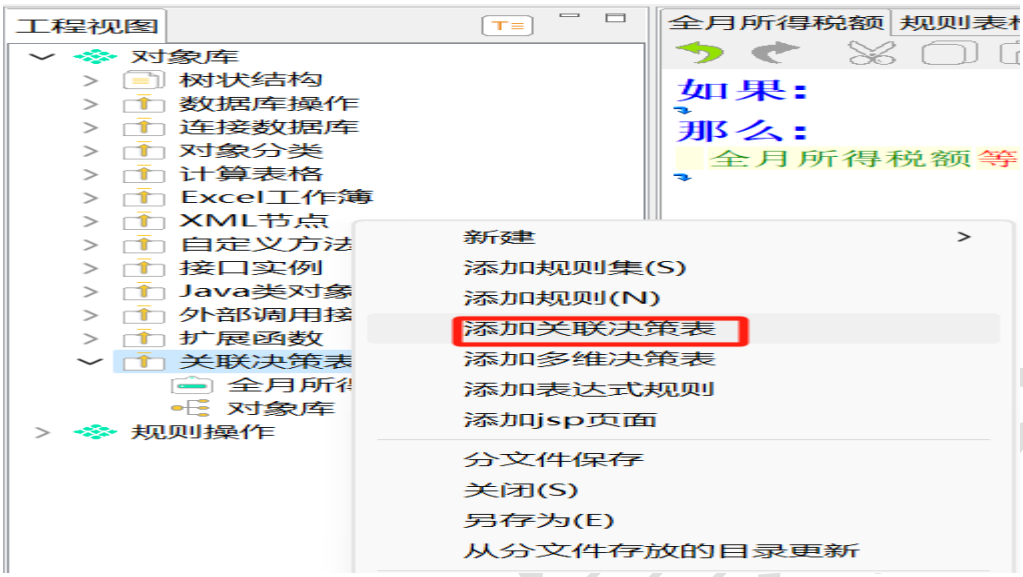
创建完成后，我们要计算“全月所得税额”，在规则中我们实现的逻辑过程如下：



确认后，规则“全月所得税额”的配置工作就完成了！

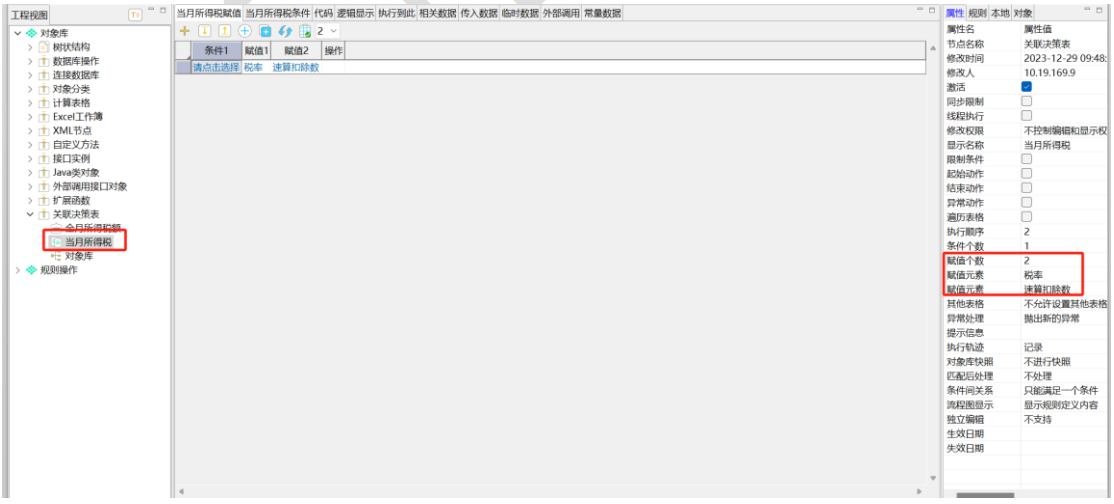
(5) 创建“关联决策表”

我们需要创建一个名为“当月所得税”的关联决策表，如下图所示：



关联决策表创建好了，我们需要修改下其属性。（因为“税率”和“速算扣除数”是根据“全月所得税额”的变化而变化的，所以这里将“全月所得税额”作为条件，“税率”和“速算扣除数”则为“赋值元素”，因此条件个数为 1，赋值个数为 2。）

在“关联决策表”的属性窗口，依次点击下拉列表将“条件个数”改为“1”、“赋值个数”改为“2”、第一个“赋值元素”选择“税率”，第二个“赋值元素”则为“速算扣除数”。其结果如下：

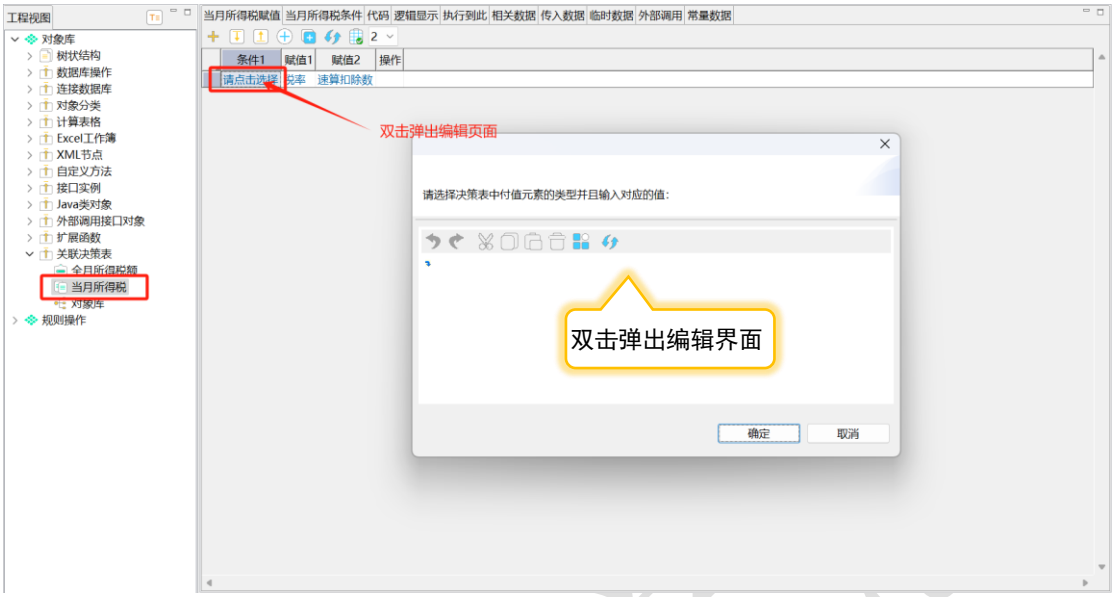


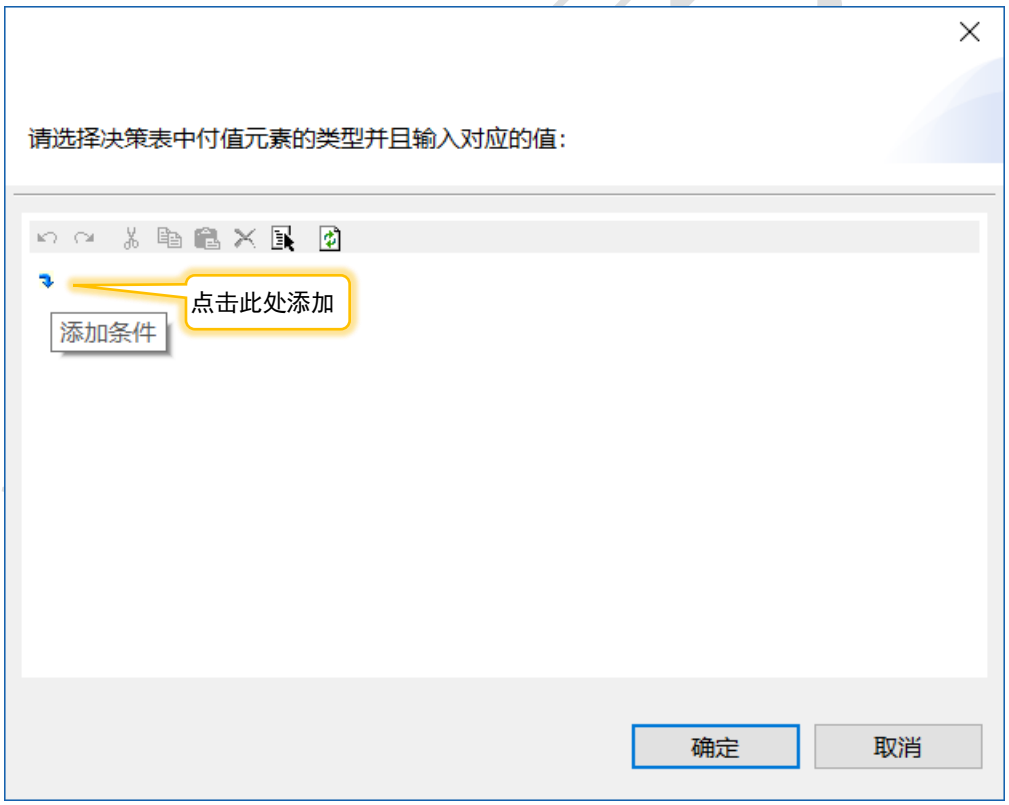
“关联决策表”的属性修改完成之后，我们需要给关联决策表配置逻辑，有两处可以进行条件设置：

第一处是：在“当月所得税赋值”下面的工具栏点击 或是双击条件下面的 请点击选择，在弹出界面进行添加；

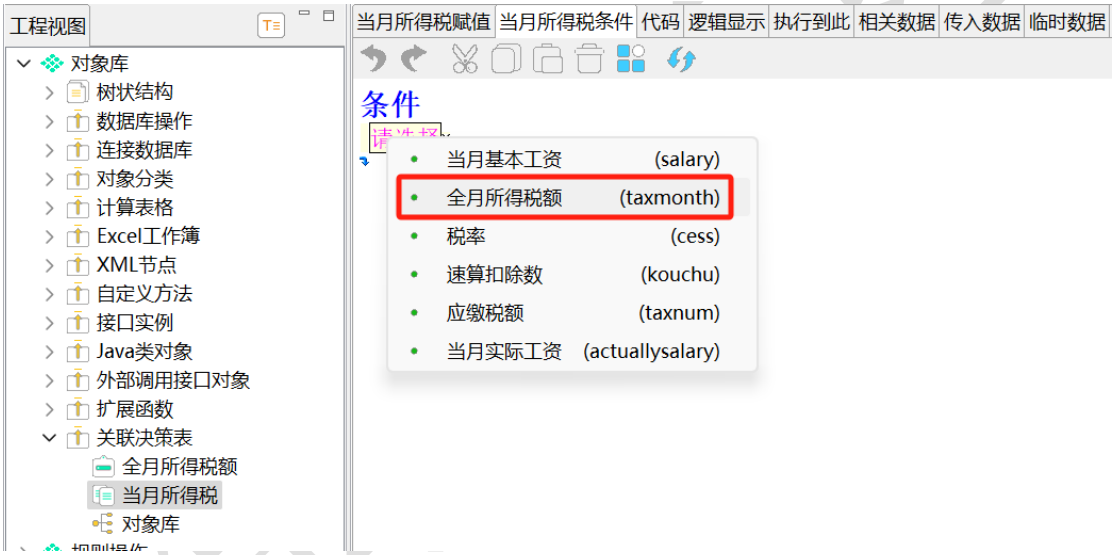
第二处是：切换至“当月所得税条件”界面进行添加。

两处编辑的操作、结果是一样的。若使用第一种方式来实现，其步骤如下图所示：

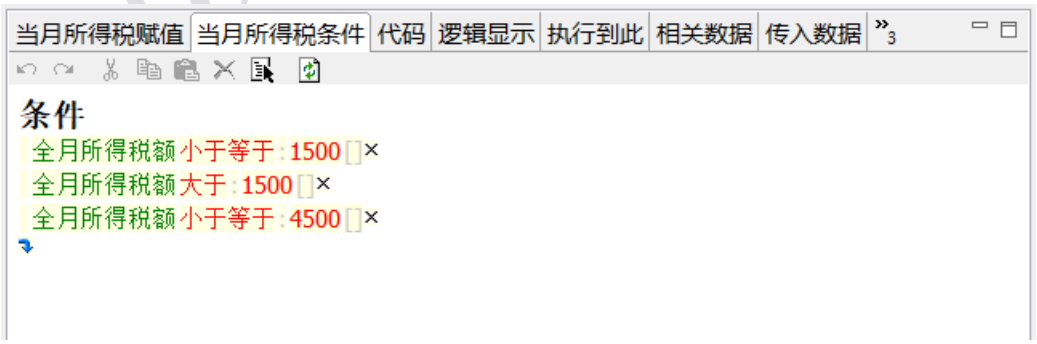




若采用第二种，步骤如下：

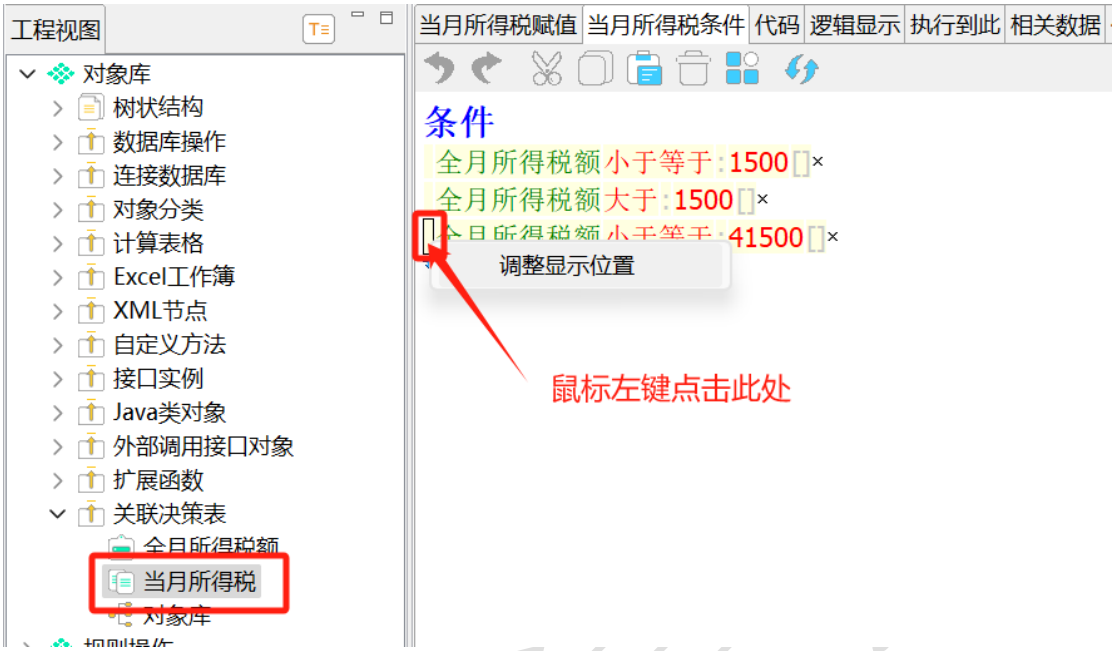


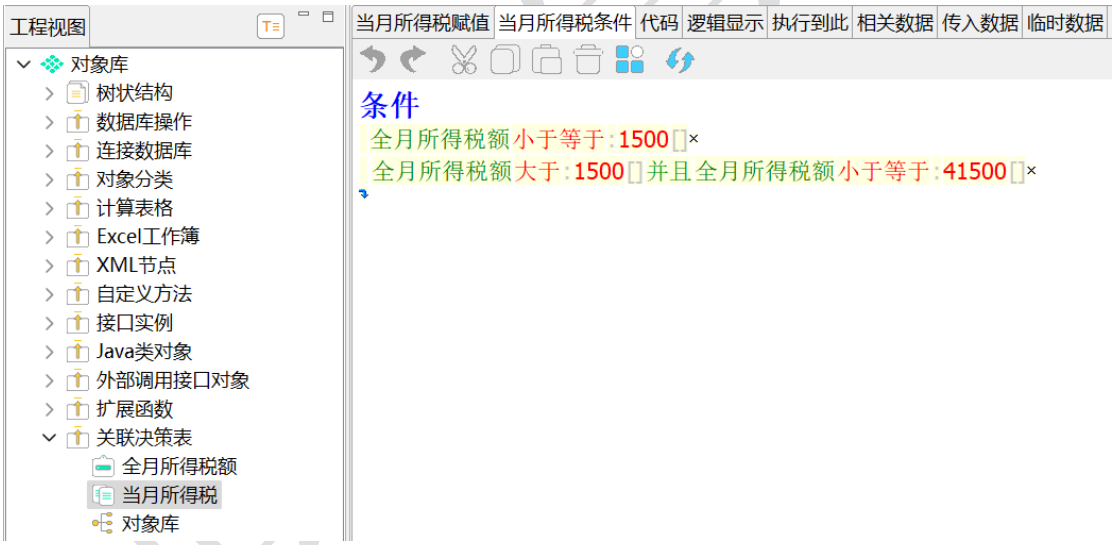
无论采用上述的那哪方式“添加条件”。在点击了“添加条件”之后，我们就可以设置“关联决策表”的条件了，如下图所示：



在上图中“全月所得税额 大于 1500”、“全月所得税额 小于等于 4500”组成了一个范围区间，类似于（1500 < i <= 4500），所以这里我们把这两个条件合并为一个条件。

点击“全月所得税额”前面的来“设置复合条件”：





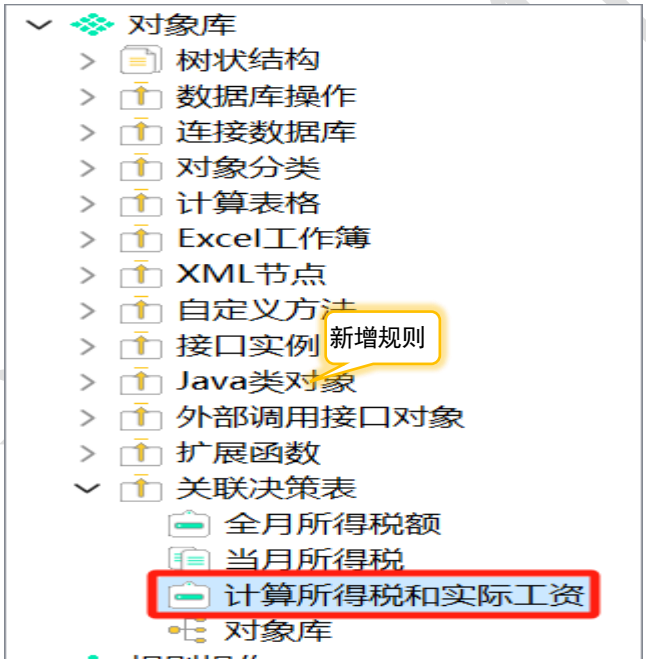
完成后，条件如下图所示：

然后我们要根据“全月所得税额”，设置相应的“税率”和“速算扣除数”如下所示：

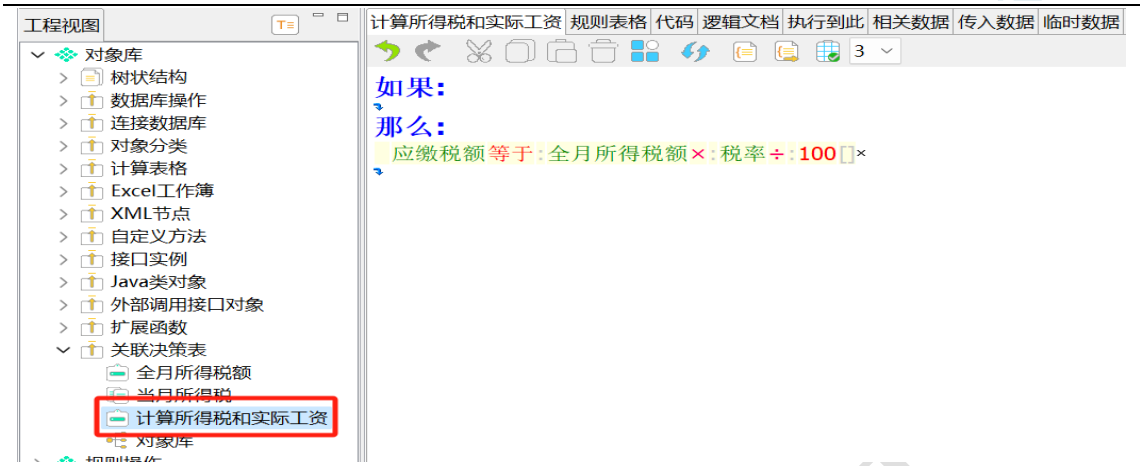
(6) 创建“计算所得税和实际工资”规则

我们从关联决策表中，可以得到“全月所得税额”对应的税率和速算扣除数。在这一规则中，我们就可以根据税率和速算扣除数，计算到“应缴税额”和“当月实际工资”。

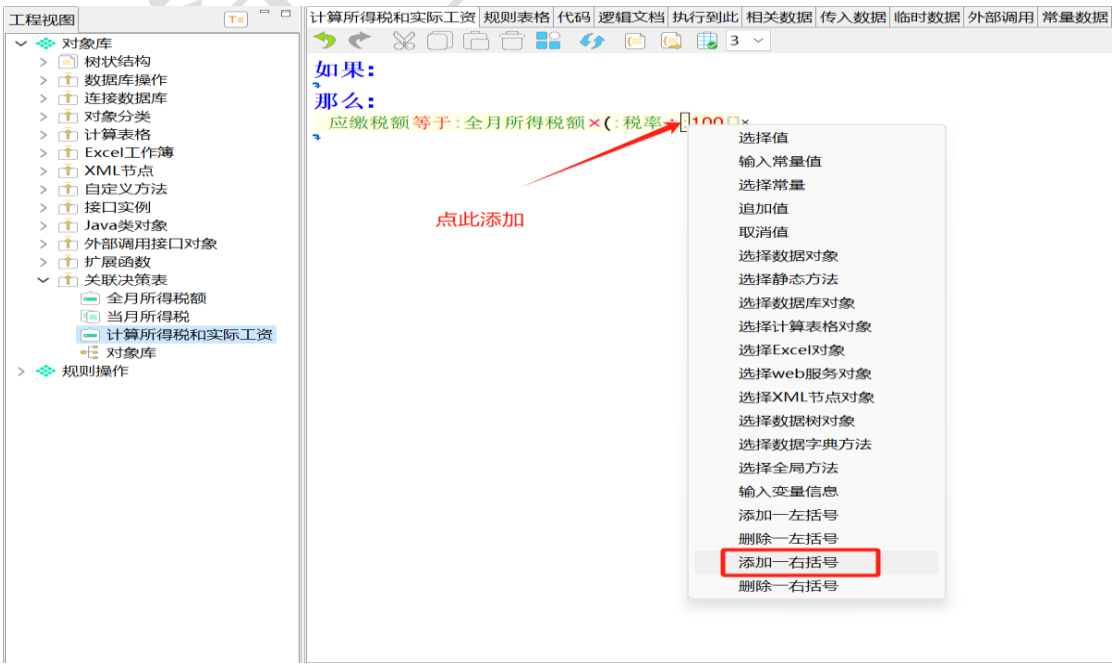
添加“计算所得税和实际工资”的规则，如下图所示：

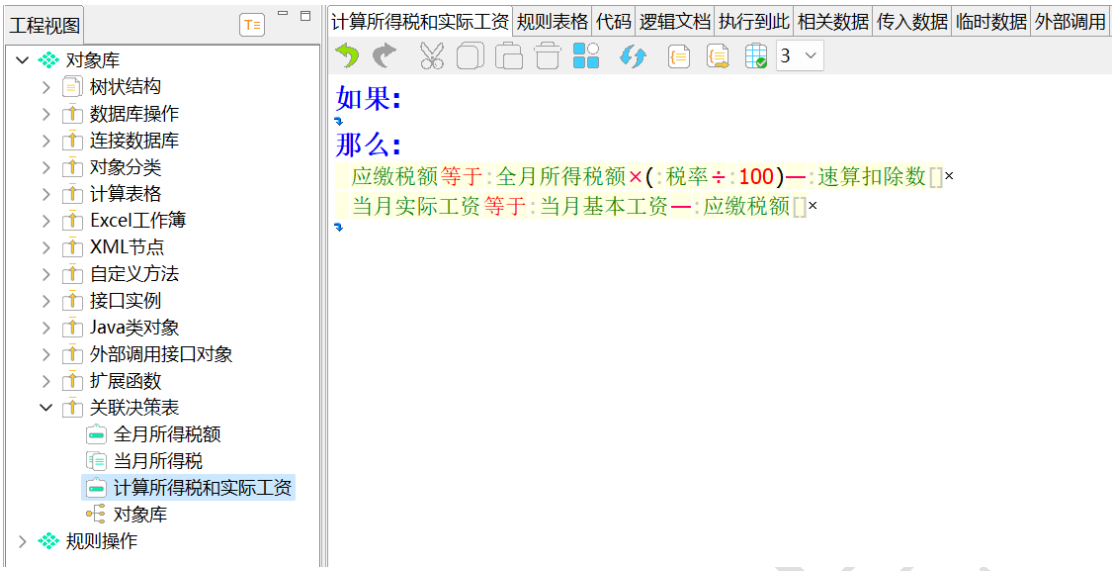


添加完成后，我们需要添加条件，过程如下：



由于在“关联决策表”中税率设置的是整数，所以在规则中设置税率时需要除以 100，并在“税率”之前和“100”之后加上括号，如下图：





(7) 保存并编译

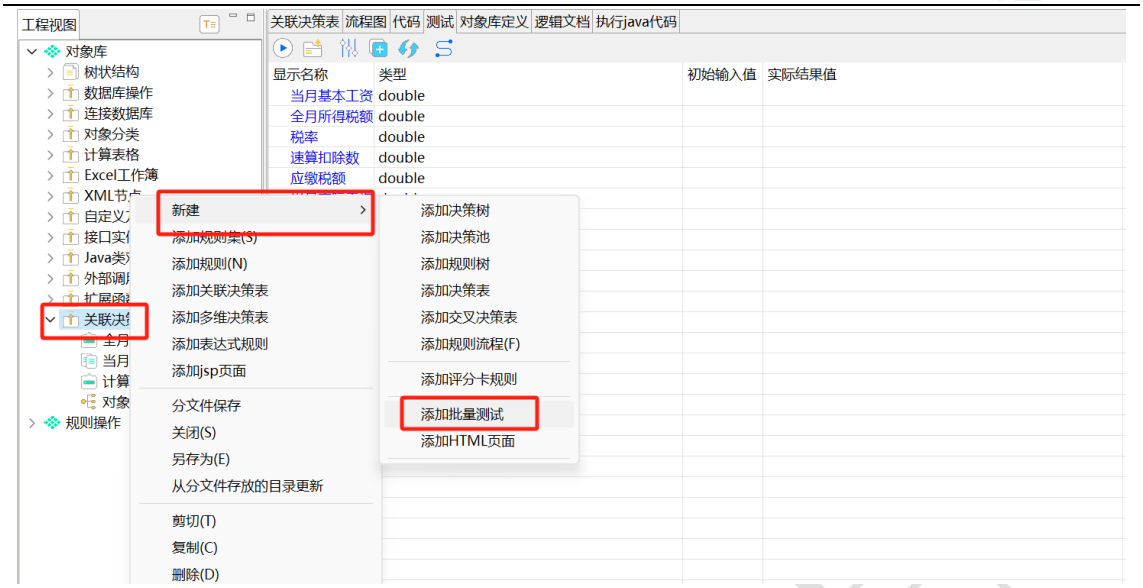
规则逻辑完成后，我们需要对规则进行保存、编译，步骤如下图所示：

点击“全部保存”按钮，然后在消息窗口会有如下图所示的记录：

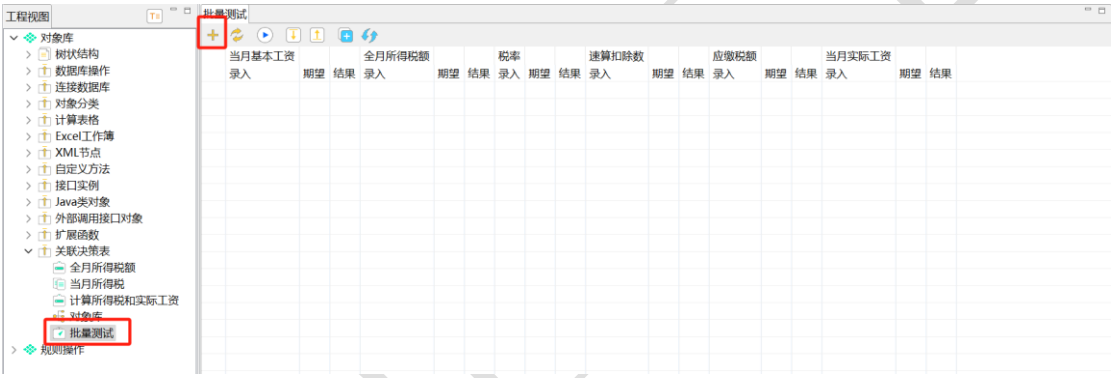
这两句消息记录表[illegible]编译。若该规则包出现错误，在消息窗口将会出现红色字体的提示。

3. 测试

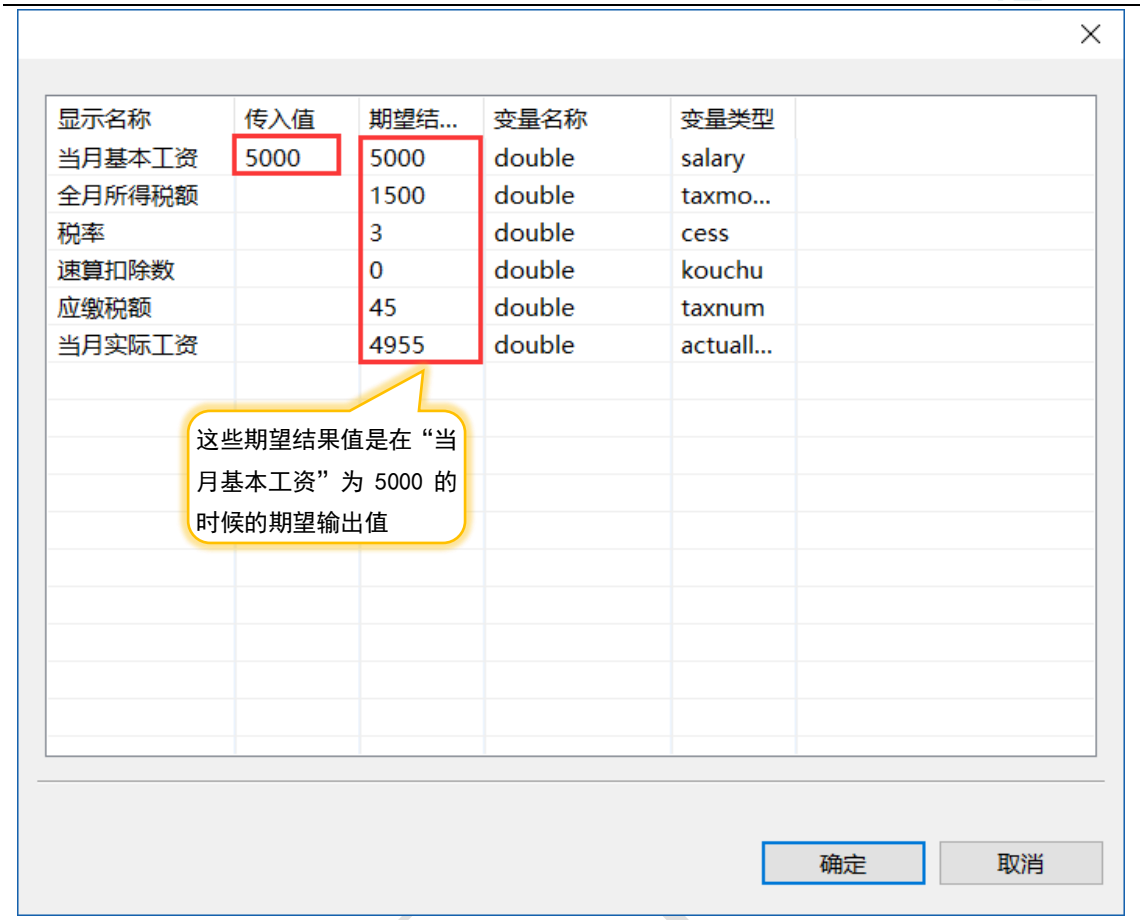
在规则包编写完成，保存、编译之后，我们需要测试该规则包的输入，输出结果值是否正确。首先，我们为该规则包添加一个“批量测试”，步骤如下图所示：



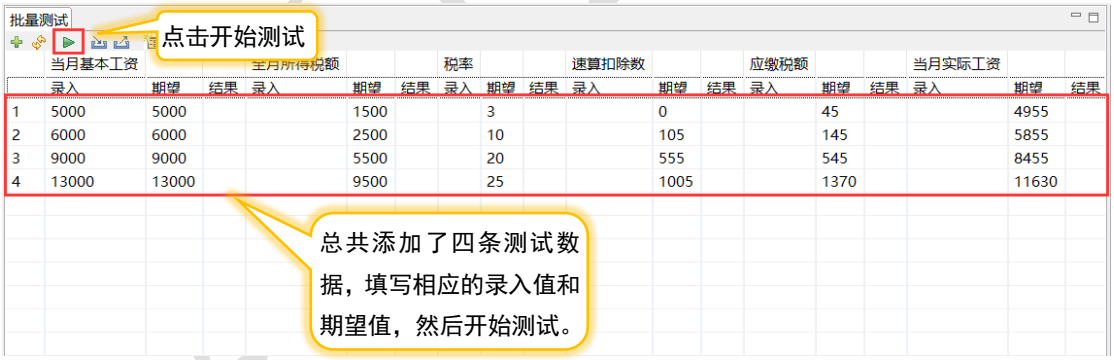
添加完成后，如下图所示：



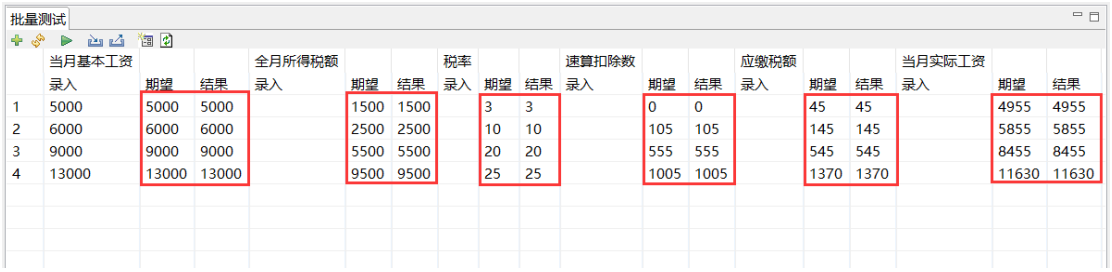
然后我们需要给该批量测试添加若干个测试用例，点击上图中黑色框中的“ ”按钮，弹出的界面如下所示，其中“传入值”是输入值，“期望结果值”是在该输入值下希望输出结果值，若“期望结果值”与实际输出结果值相同，则说明该条测试数据测试成功！



再次点击“ ”添加多条测试数据，完成后结果如下：



点击了“测试”按钮之后，可以在“批量测试”中查看测试输出的实际结果值。



期望值与结果值相等，测试成功，程序执行正确。

3.3. 交差决策表

1. 业务需求

现需解决每个员工每个月（上半年）的基本业务工资。由于每个员工每个月都有相对应的工资，同样的每个月也对应着每个员工，他们之间存在一一对应的关系。这里我们就可以设计个“交叉决策表”，把月份和员工分别作为“交叉决策表”的横向和纵向条件，把工资作为“交叉决策表”的赋值元素。需要注意的是：交叉决策表的赋值元素只能有一个!

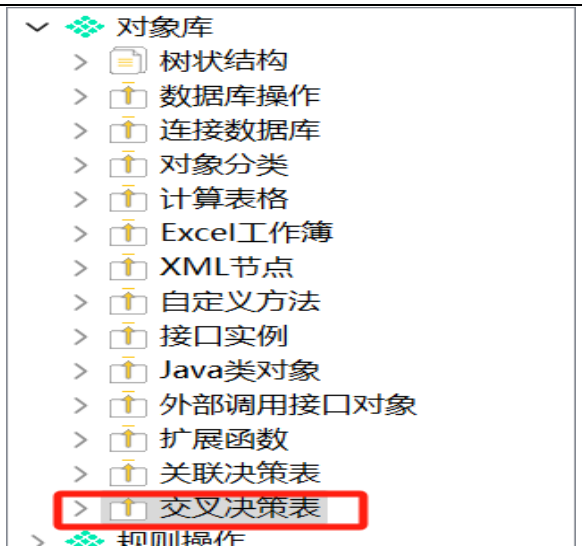
员工的工资清单如下：

工资	月份					
员工姓名	1	2	3	4	5	6
小王	4000	4600	4401	4300	4500	4300
小张	5000	5600	5550	5100	5400	5600
小李	7000	7800	7400	7300	7500	7600

2. 规则实现

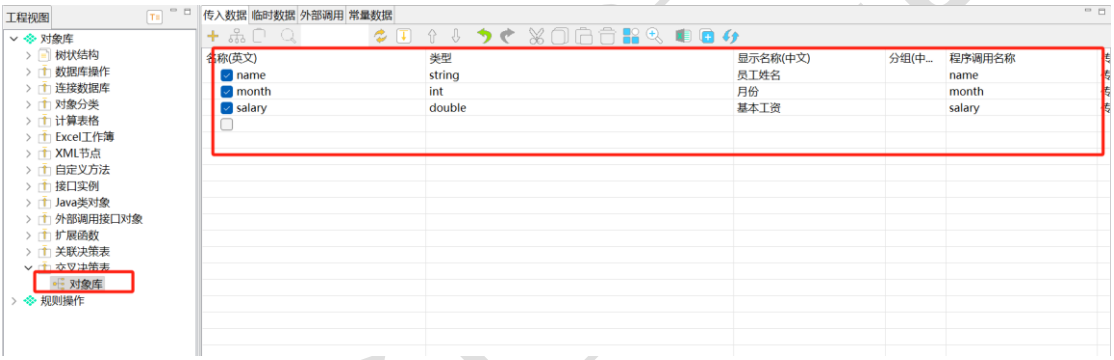
(1) 规则包创建

右键名为“规则操作”的工程，点击“新建规则包”，创建一个名为：“交叉决策表”的规则包，如下图所示：



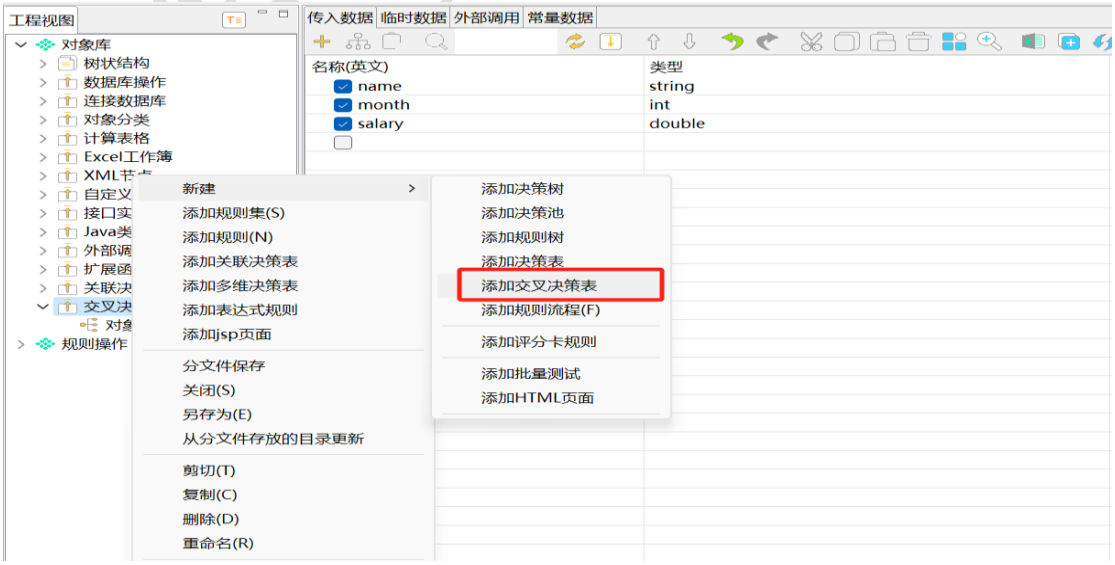
(2) 变量定义

该规则包中需要在对象库中，定义三个变量：员工姓名（name），月份（month），基本工资（salary）。如下图所示：



(3) 创建交叉决策表

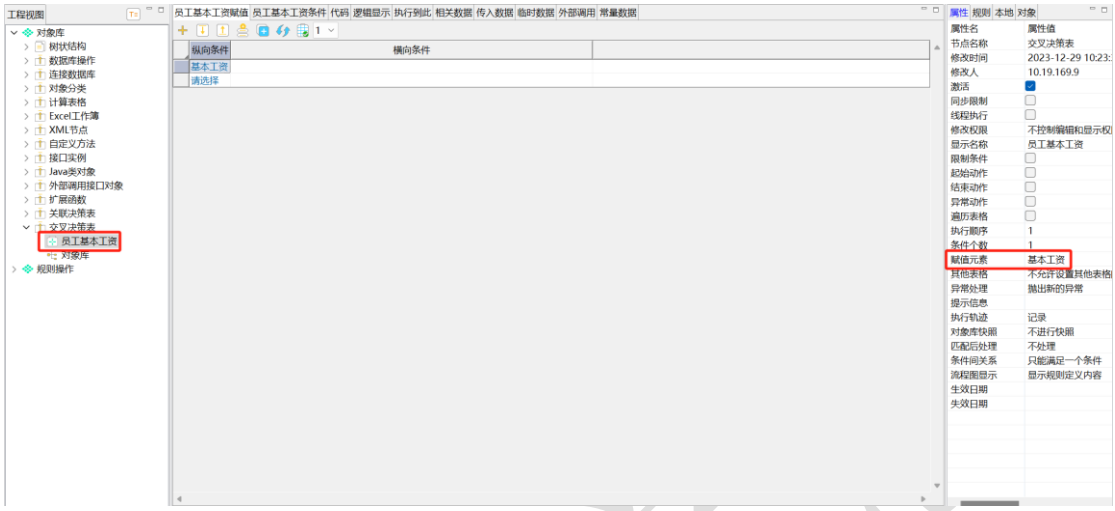
右键名为“交叉决策表”的规则包，选择“添加交叉决策表”。添加完成后，需将该决策表重命名为“员工基本工资”，操作截图如下所示：



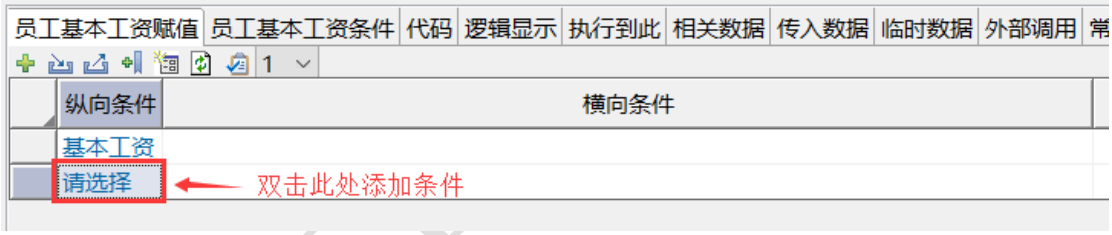
创建了“员工基本工资”的“交叉决策表”之后，需处理该决策表的业务逻辑。

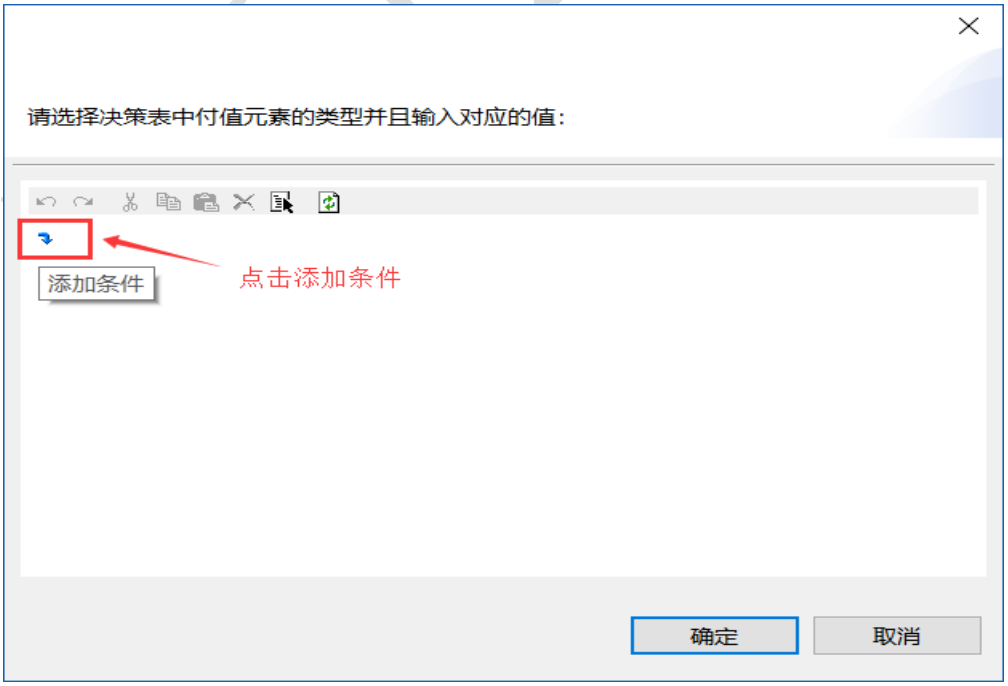
(1) 规则实现

在该例“交叉决策表”中，有两个条件：员工姓名、月份。但是在“交叉决策表”中，横向条件是缺省存在的，所以在该“交叉决策表”的属性窗口只需一个纵向条件，然后在“赋值元素”中选择“基本工资”。设置完成后，“交叉决策表”的属性如下图所示：

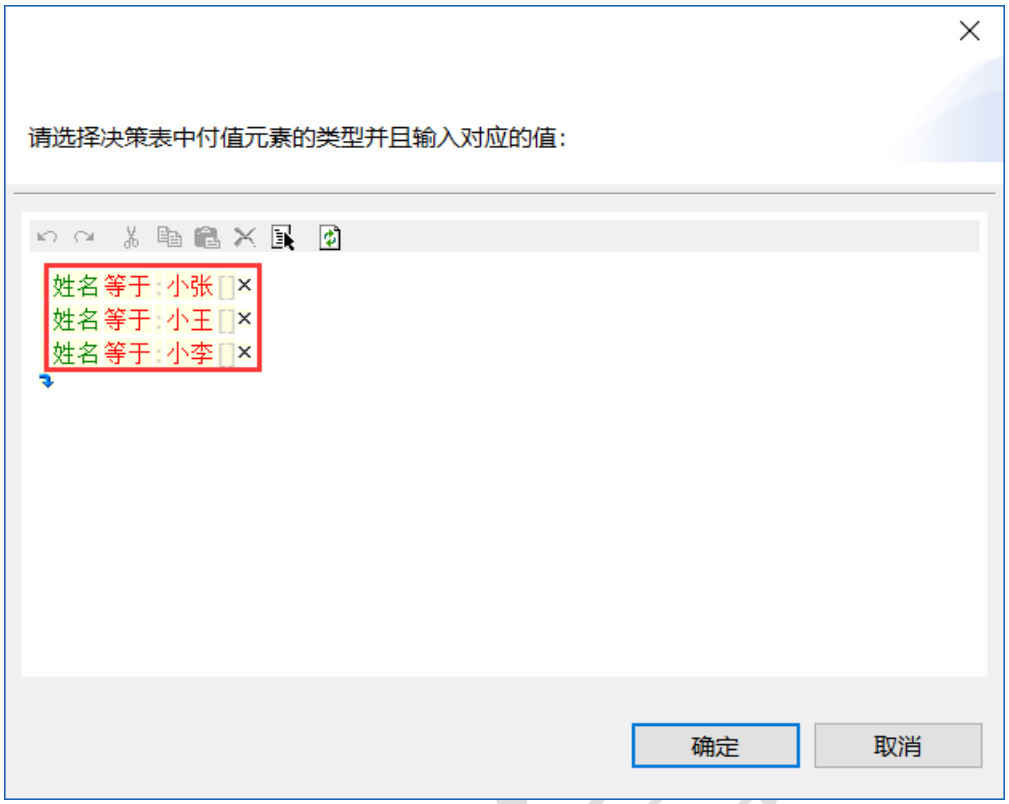


属性设置完成后，我们要为该交叉决策表设置对应的条件，操作步骤如下图所示：

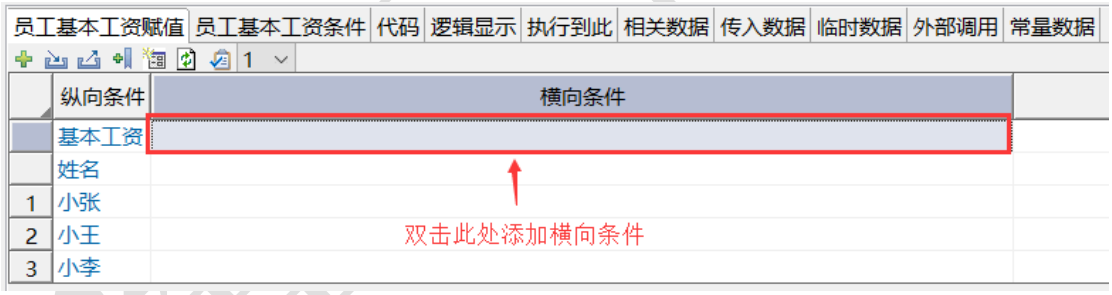




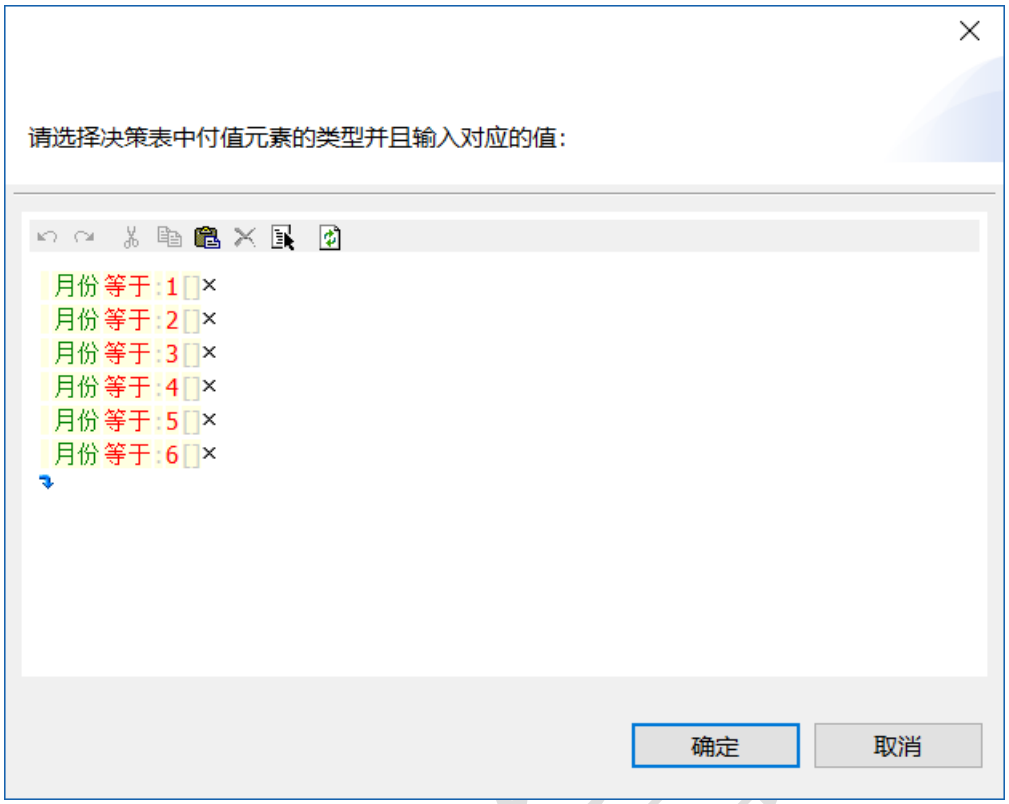
在弹出框中添加以下条件：



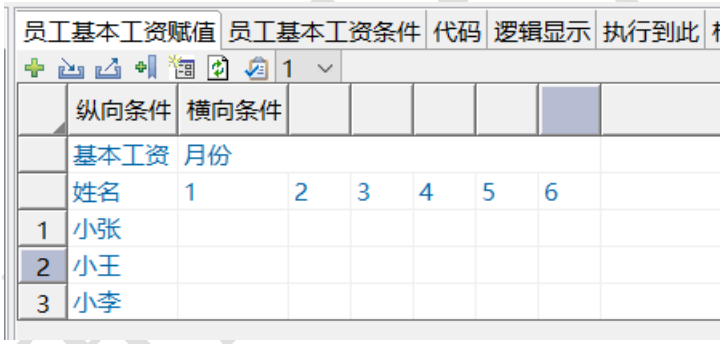
员工姓名设置完成以后，我们还要在条件部分添加 6 种月份信息。双击“基本工资”右下方的方框区域，设置如下图所示的逻辑，步骤如下图所示：



在弹出框中添加如下条件：

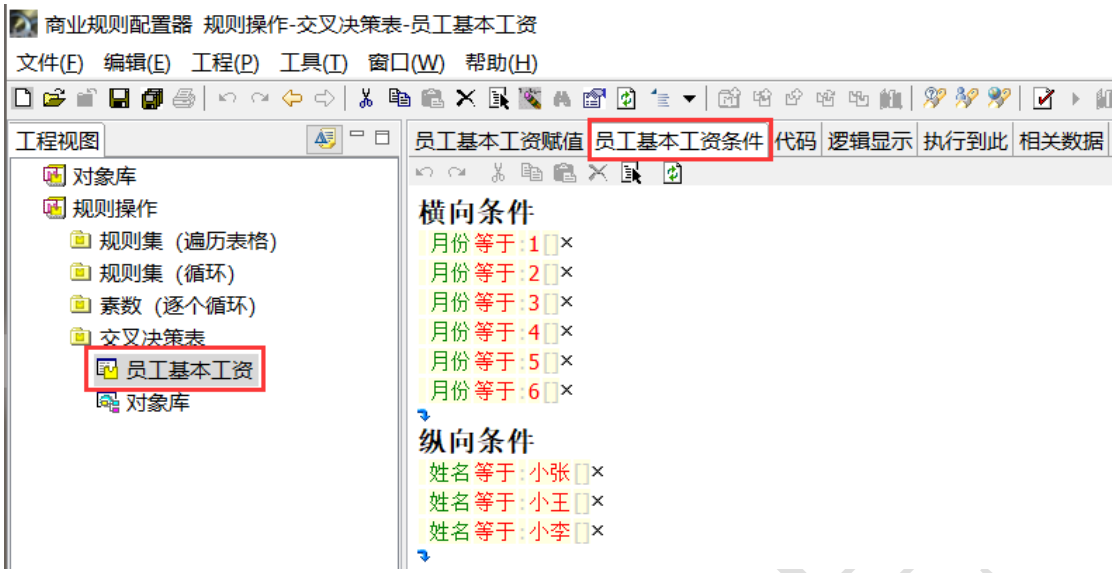


确定后可以看到如下图所示页面：



最后我们依次修改每个员工，每个月份的的工资，完成的结果如下：

点击编辑窗口中的“员工基本工资条件”选项卡，可以看到具体的规则逻辑。如下图所示：



注：“交叉决策表”的横向条件缺省存在的且只能有一个；纵向条件可以设置多个，在每个纵向条件中可以存在多个不同的子条件，最后所有的的纵向条件和横向条件交叉组成新条件。

3. 测试

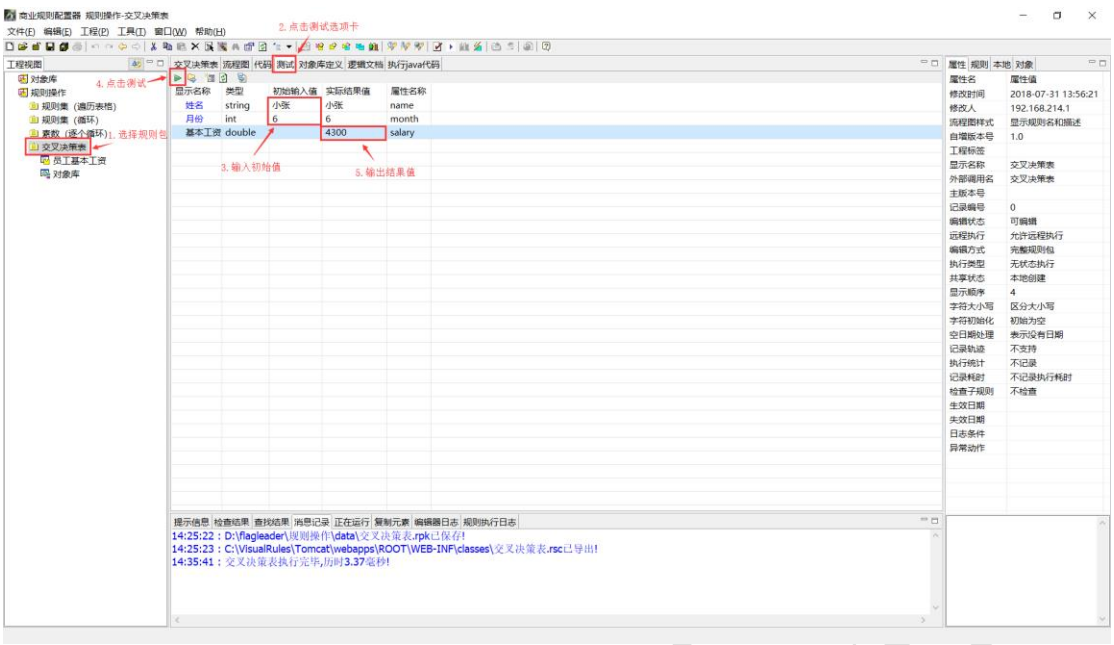
我们在规则编写完成之后，都需为规则进行保存、编译。点击“全部保存”，配置器就会自动保存并编译所有未保存规则包，过程如下：

保存后，我们可以在消息窗口看到如下消息：

如果出现上述的两条信息，说明该规则包已经编译成功！规则包保存编译以后，我们需要测试该规则包的输出结果正确性。规则包的测试有两种方式：一种是批量测试、另一种是规则单元测试。

(1) 规则单元测试

规则单元测试一次只能测试一条数据，若需测试其他情况，需要修改初始输入值，再测试一遍。测试的步骤和结果，按照下图所示的进行：



3.4. 多维决策表

1. 业务需求

在交叉决策表以及关联决策表中，条件之间的通常是一对一的关系（也可以实现一对多），但是在实际情况中往往会出现一对多的关系。如在下面的列子中，一个学生要考很多学科，一个学期又要考很多场试。若用交叉决策表会造成逻辑上的冗余，而多维决策却很容易的实现一对多的关系，。学生考试的考试情况如下图所示：

	A	B	C	D
1	学生姓名	考试	学科	得分
2	小张	期中考试	语文	88
3	小张	期中考试	数学	89
4	小张	期中考试	英语	87
5	小张	期末考试	语文	89
6	小张	期末考试	数学	86
7	小张	期末考试	英语	90
8	小张	模拟考试	语文	90
9	小张	模拟考试	数学	88
10	小张	模拟考试	英语	86
11	小王	期中考试	语文	90
12	小王	期中考试	数学	99
13	小王	期中考试	英语	91
14	小王	期末考试	语文	92
15	小王	期末考试	数学	98
16	小王	期末考试	英语	89
17	小王	模拟考试	语文	93
18	小王	模拟考试	数学	96
19	小王	模拟考试	英语	87
20	小李	期中考试	语文	80
21	小李	期中考试	数学	83
22	小李	期中考试	英语	78
23	小李	期末考试	语文	81
24	小李	期末考试	数学	85
25	小李	期末考试	英语	82
26	小李	模拟考试	语文	88
27	小李	模拟考试	数学	85
28	小李	模拟考试	英语	86

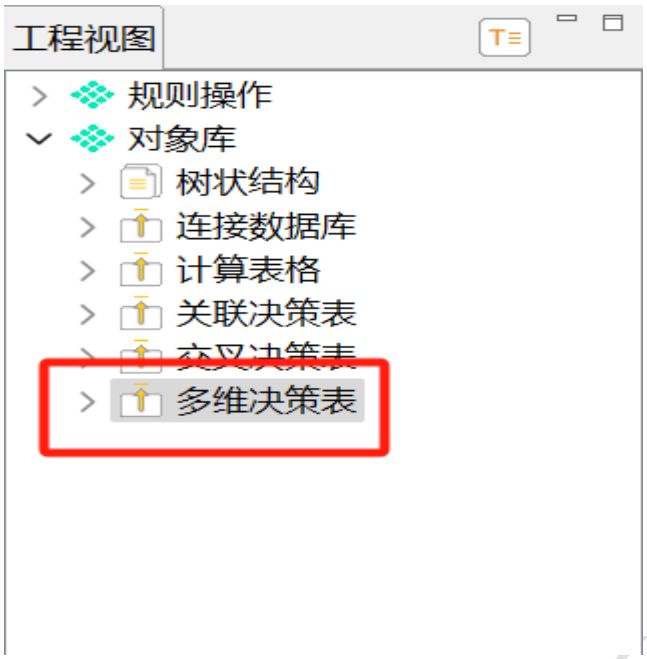
我们可以看到，每个学生每学期要有三次考试，而每次考试要考三门学科。这样多维决策表的条件部分应该有三个：学生姓名、考试类型、学科。而结果只有一个：分数。

需要注意的是：虽然多维决策表可以实现多对多的关系，但是在每个条件之间必须公用同一个条件。例如，在本例子中若实际情况中有的学生没有学习英语，但是在多维决策表中还是会有该学生的英语成绩的。若要实现每个条件下的子条件不同，就要用交叉决策表来实现了。

2．规则实现

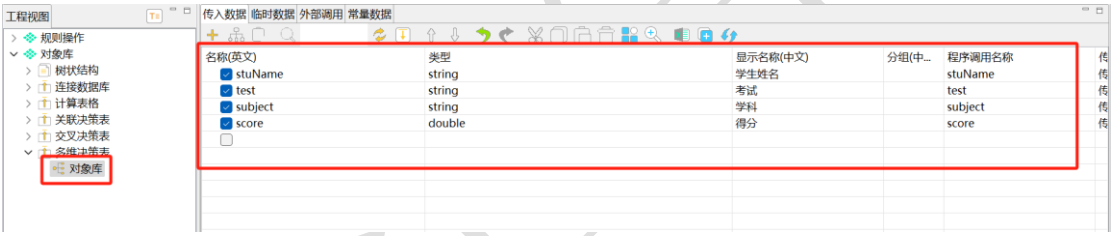
(1) 规则包创建

右键名为“规则操作”的工程，点击“新建规则包”，创建一个名为“多维决策表”的规则包。如下图所示：



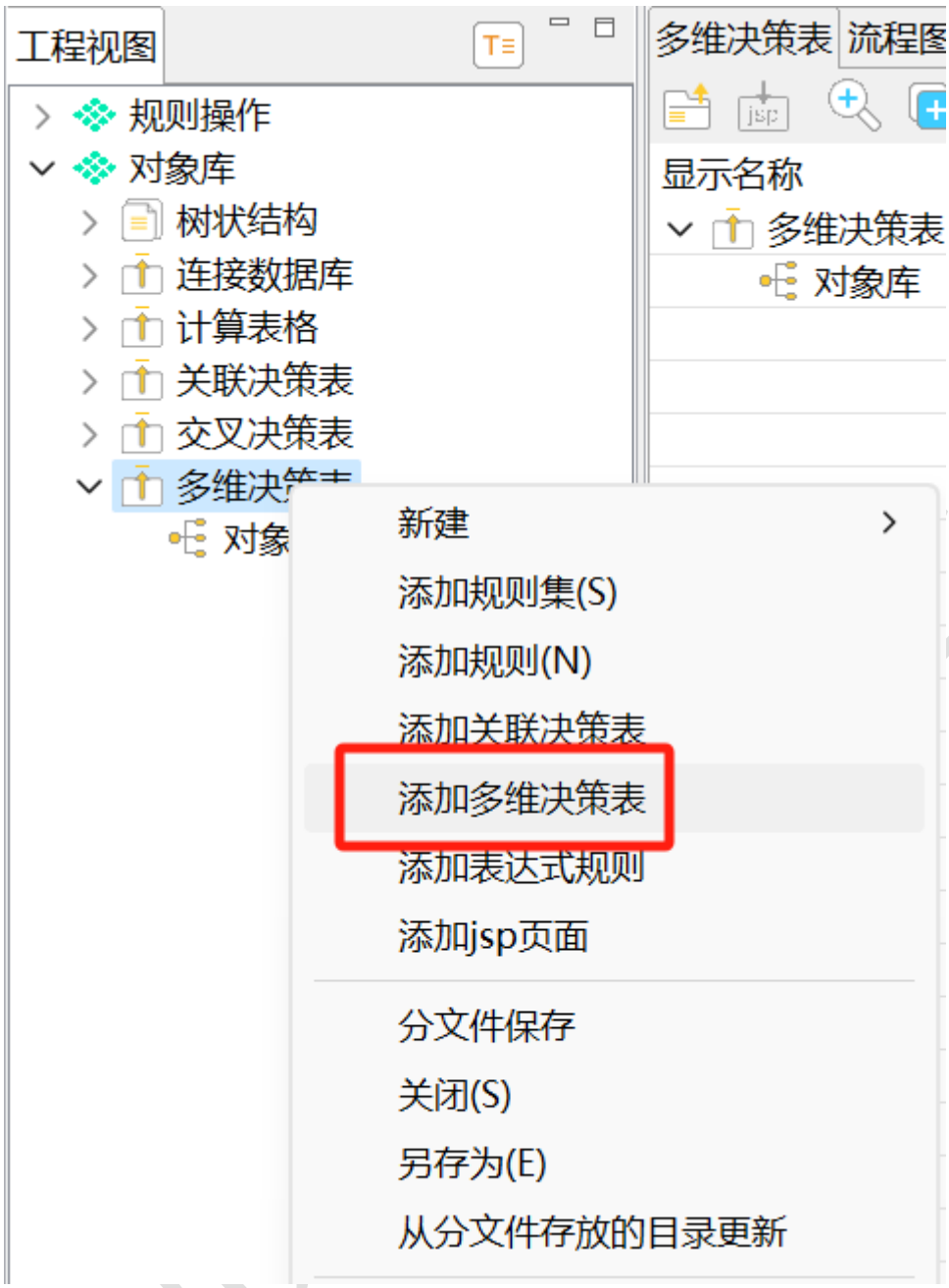
(2) 定义变量

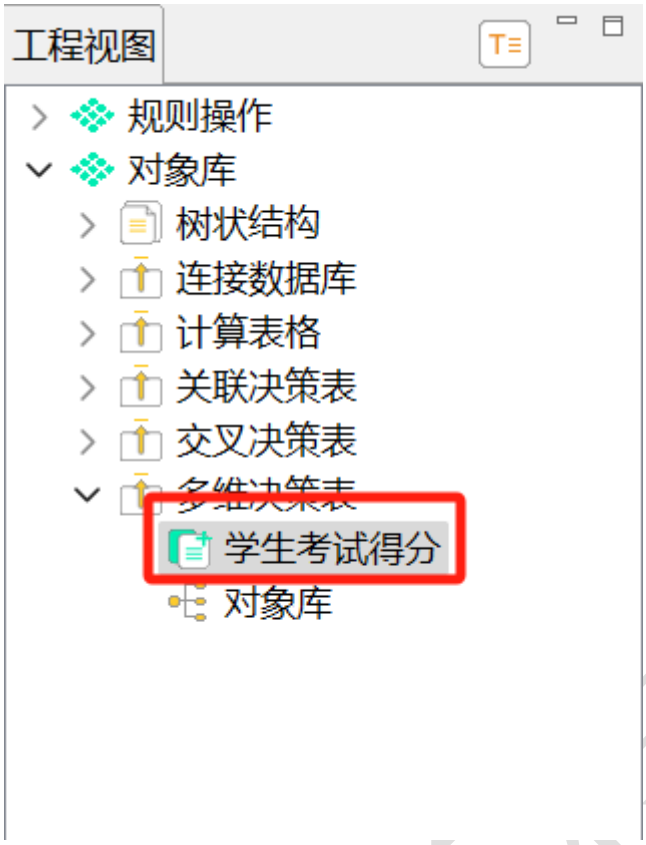
需要在该规则包的对象库中，定义四个变量：学生姓名（stuName），考试（test），学科（subject），得分（score）。如下图所示：



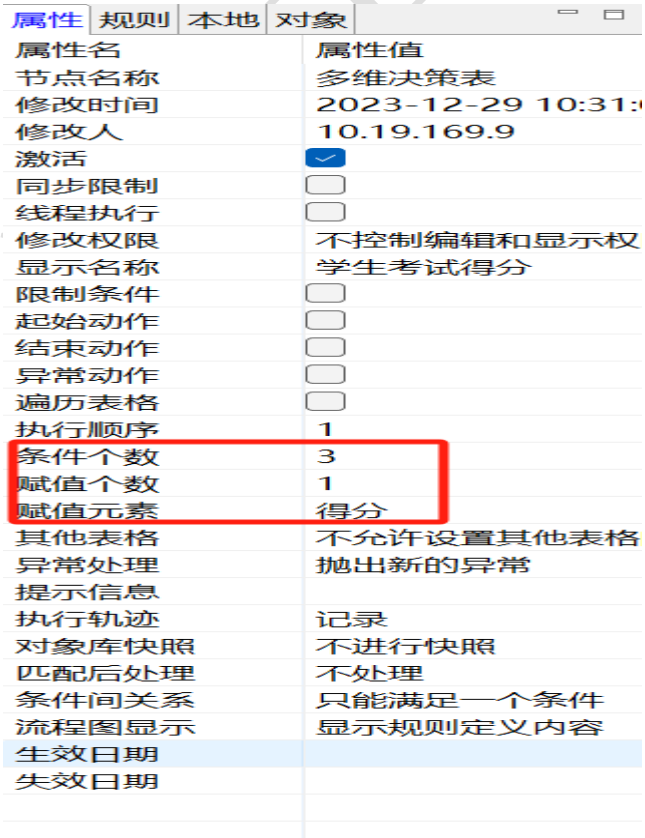
(3) 逻辑实现

点击“多维决策表”规则包，右键创建名为“学生考试得分”的多维决策表，创建过程如下图所示：


工程视图
规则操作
对象库
树状结构
连接数据库
计算表格
关联决策表
交叉决策表
多维决策表
对象
多维决策表
流程图
显示名称
多维决策表
对象库
新建
添加规则集(S)
添加规则(N)
添加关联决策表
添加多维决策表
添加表达式规则
添加jsp页面
分文件保存
关闭(S)
另存为(E)
从分文件存放的目录更新

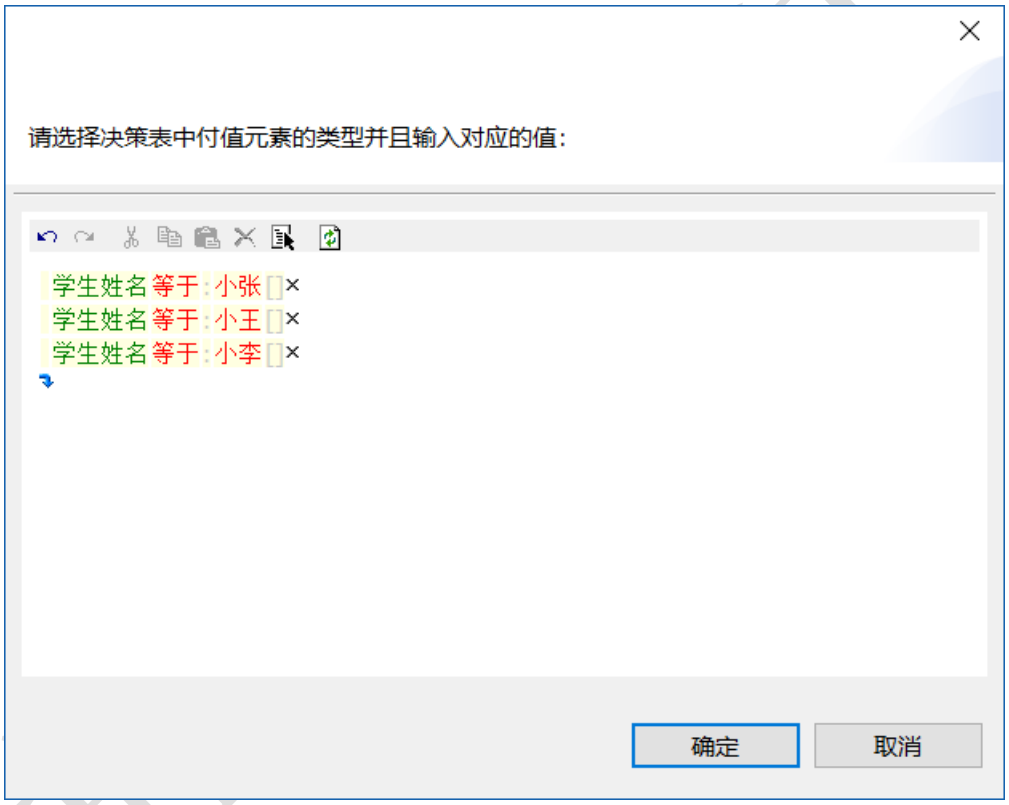


创建好了“多维决策表”我们需要设置其属性，首先要在属性窗口，把条件个数设置为3，赋值元素设置成“得分”（score）。操作流程如下所示：

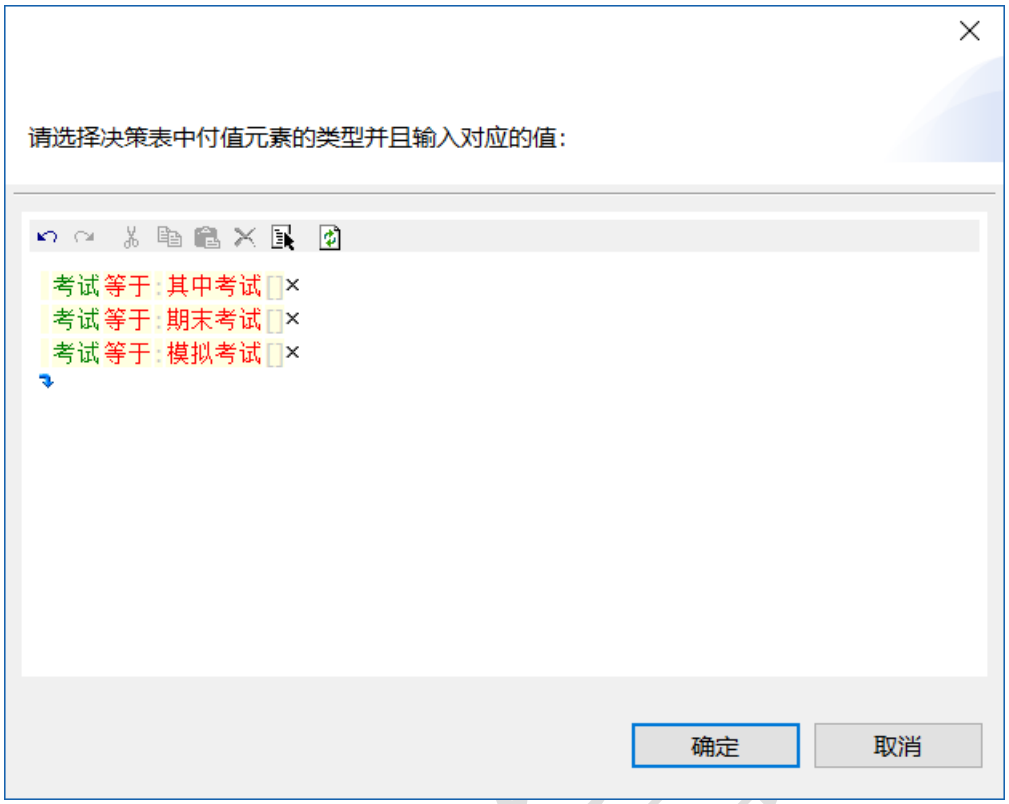


属性设置好了之后，我们要在 “多维决策表”的条件部分中设置具体的逻辑以及该条件下的“得分”。条件设置过程如下：

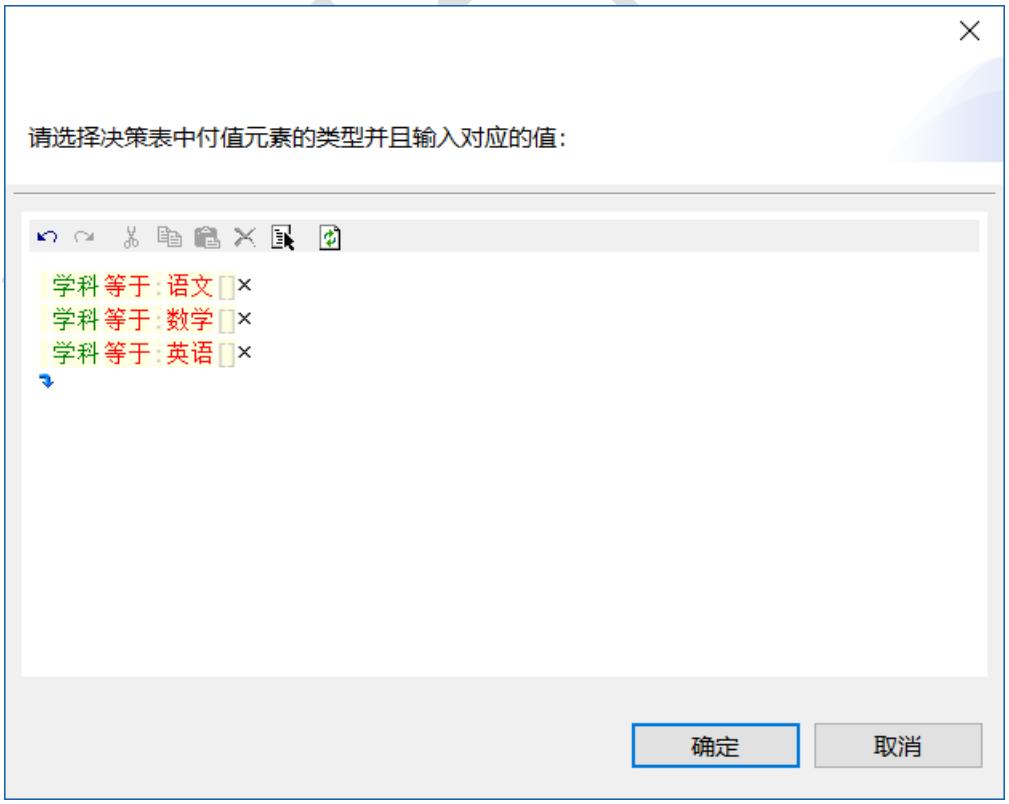
在弹出框中添加如下条件：



这样学生姓名就设置好了，然后我们再添加考试信息，完成后如下图：



考试信息的条件设置完成之后，我们还要给学科添加信息，其添加完成后如下图所示：



学科的条件设置完成之后，这样多维决策表的条件部分就完成。多维决策表的条件，如

下图所示：

学生考试得分赋值 | 学生考试得分条件 | 代码 | 逻辑显示 | 执行到此 | 相关数据 | 传入数据 | 临时数据 | 外部调用 | 常量数据

	条件1	条件2	条件3	赋值1
	学生姓名	考试	学科	得分
1	"小张"	"期中考试"	"语文"	""
2			"数学"	""
3			"英语"	""
4		"期末考试"	"语文"	""
5			"数学"	""
6			"英语"	""
7		"模拟考试"	"语文"	""
8			"数学"	""
9			"英语"	""
1..	"小王"	"期中考试"	"语文"	""
1..			"数学"	""
1..			"英语"	""
1..		"期末考试"	"语文"	""
1..			"数学"	""
1..			"英语"	""
1..		"模拟考试"	"语文"	""
1..			"数学"	""
1..			"英语"	""
1..	"小李"	"期中考试"	"语文"	""
2..			"数学"	""
2..			"英语"	""
2..		"期末考试"	"语文"	""
2..			"数学"	""
2..			"英语"	""
2..		"模拟考试"	"语文"	""
2..			"数学"	""
2..			"英语"	""

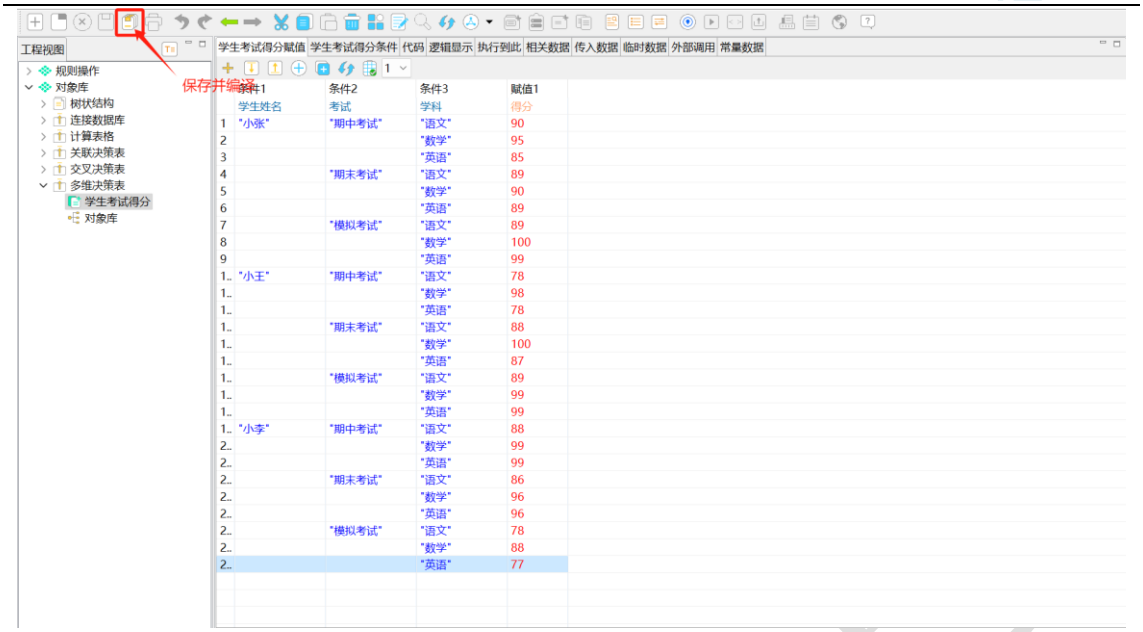
最后，我们要为每个学生的每次考试中的每个学科，赋予相应的分数。如下图：

学生考试得分赋值 | 学生考试得分条件 | 代码 | 逻辑显示 | 执行到此 | 相关数据 | 传入数据 | 临时数据 | 外部调用 | 常量数据

	条件1	条件2	条件3	赋值1
	学生姓名	考试	学科	得分
1	"小张"	"期中考试"	"语文"	90
2			"数学"	95
3			"英语"	85
4		"期末考试"	"语文"	89
5			"数学"	90
6			"英语"	89
7		"模拟考试"	"语文"	89
8			"数学"	100
9			"英语"	99
1..	"小王"	"期中考试"	"语文"	78
1..			"数学"	98
1..			"英语"	78
1..		"期末考试"	"语文"	88
1..			"数学"	100
1..			"英语"	87
1..		"模拟考试"	"语文"	89
1..			"数学"	99
1..			"英语"	99
1..	"小李"	"期中考试"	"语文"	88
2..			"数学"	99
2..			"英语"	99
2..		"期末考试"	"语文"	86
2..			"数学"	96
2..			"英语"	96
2..		"模拟考试"	"语文"	78
2..			"数学"	88
2..			"英语"	77

3．保存并编译

规则逻辑完成后，我们需要对规则进行编译，步骤如下：

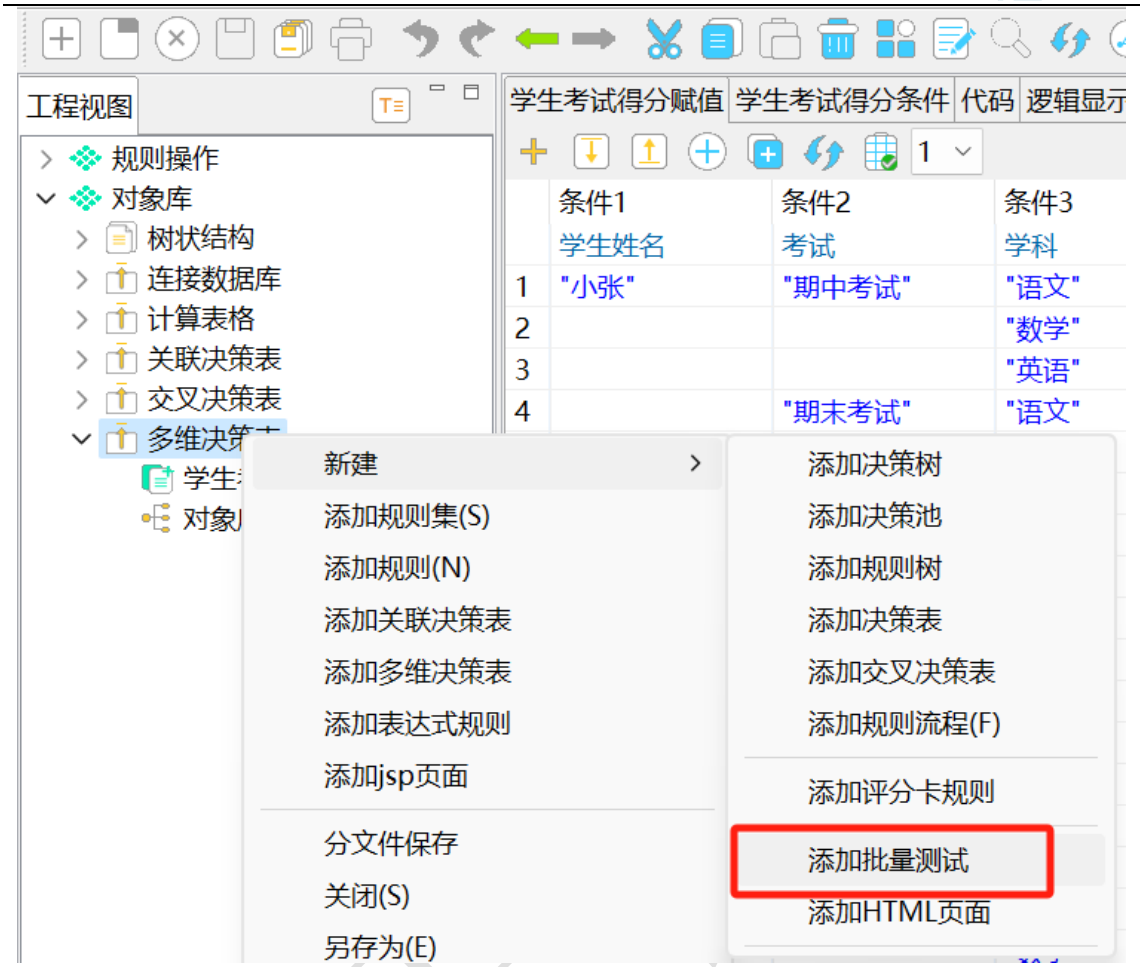


点击保存并编译后，在消息窗口出现下图所示的信息：

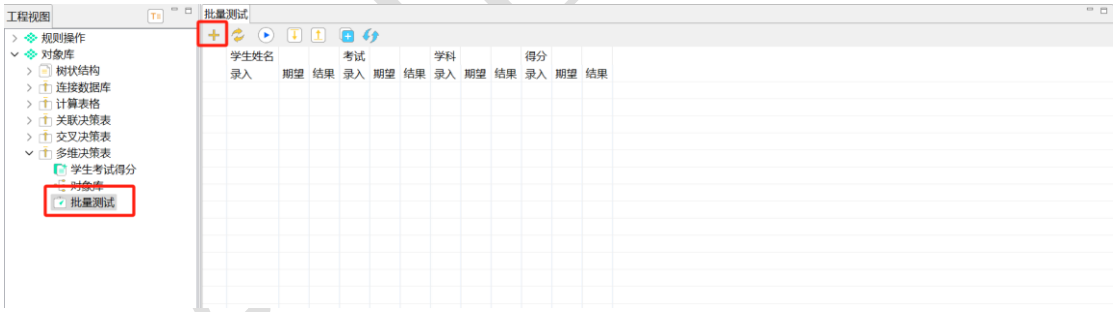
这两句话表示该规则包已成功保存并编译。若该规则包出现错误，在消息窗口将会出现红色字体的提示。

4．测试

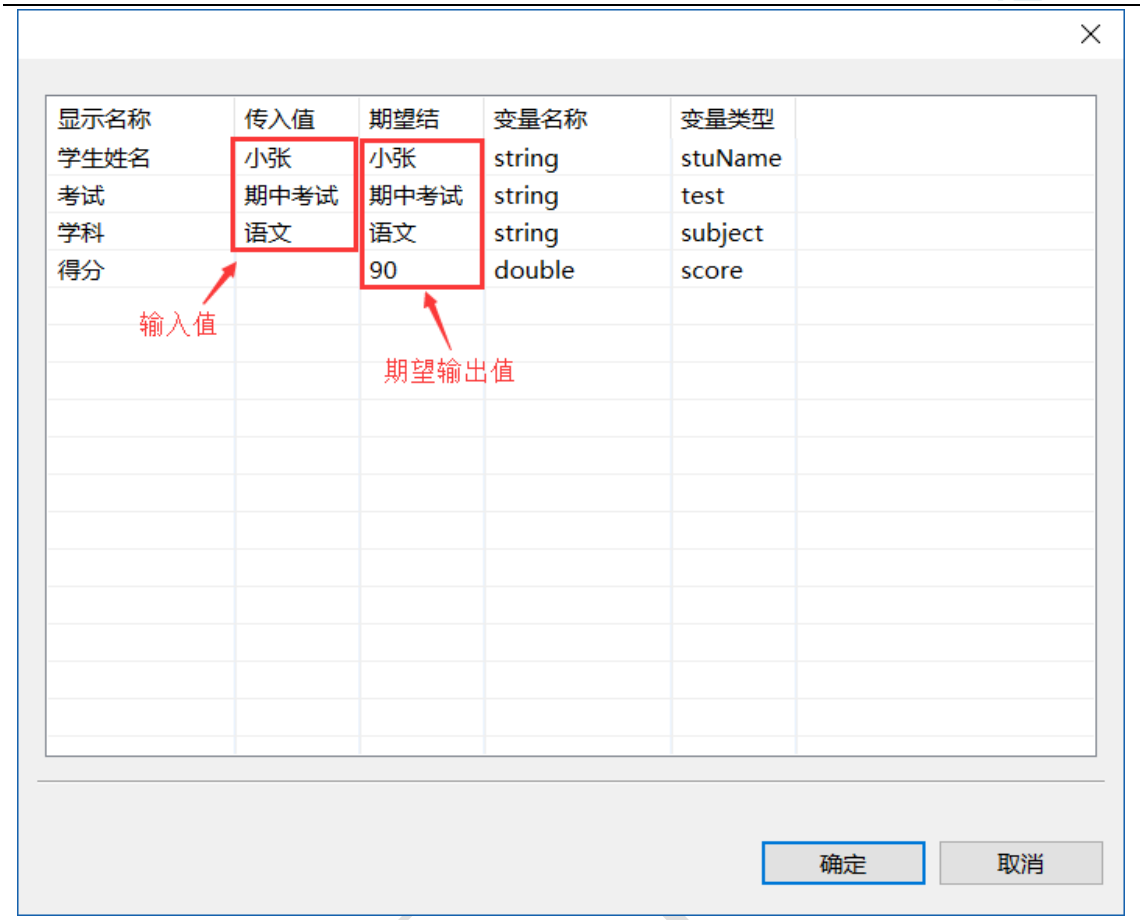
在规则包完成之后，我们需要测试该规则包的输入，输出结果值是否正确。首先，我们为该规则包添加一个“批量测试”，选择规则包，右键新建→添加批量测试：



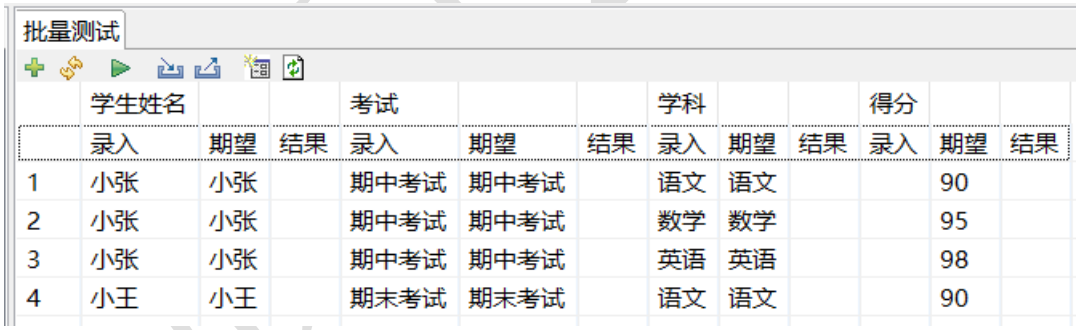
添加完成之后，选择该批量测试，在编辑窗口中点击“ ”添加测试用例：



然后我们需要给该“批量测试”添加若干个测试用例，在弹出的界面中，其中“传入值”是你输入值，“期望结果值”是在该输入值下希望输出结果值，若“期望结果值”与实际输出结果值相同，则说明该条测试数据测试成功！

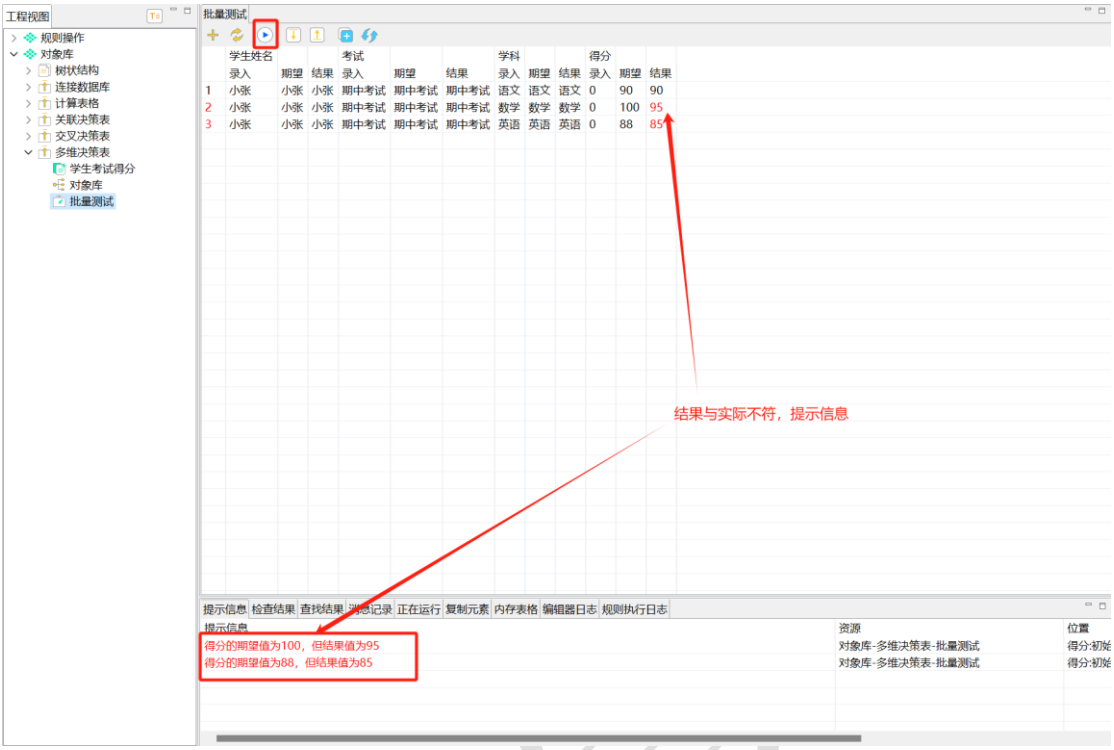


再次添加多条测试用例：



	学生姓名			考试			学科			得分		
	录入	期望	结果	录入	期望	结果	录入	期望	结果	录入	期望	结果
1	小张	小张		期中考试	期中考试		语文	语文			90	
2	小张	小张		期中考试	期中考试		数学	数学			95	
3	小张	小张		期中考试	期中考试		英语	英语			98	
4	小王	小王		期末考试	期末考试		语文	语文			90	

保存之后，点击测试按钮，分别在批量测试和消息窗口出现下图所示的提示消息：



3.5. 决策树

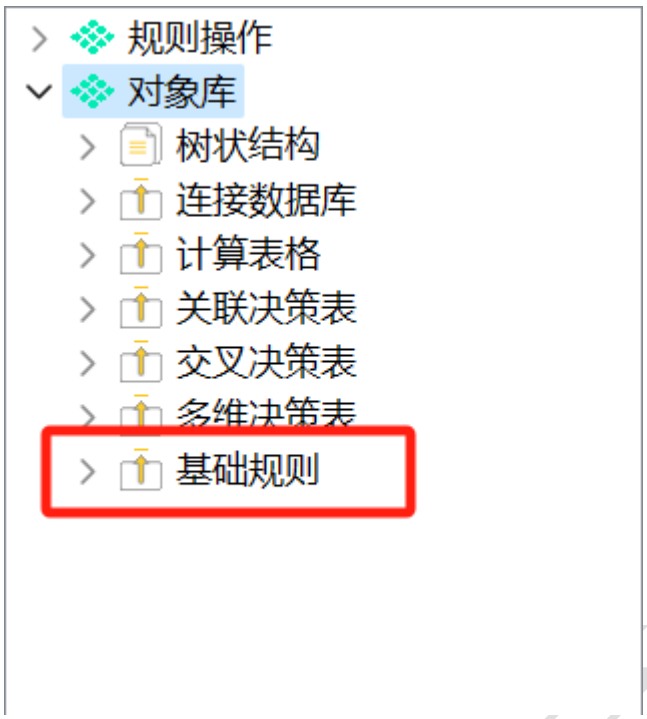
1. 功能介绍

在规则开发过程中，往往会遇到某个规则会反复被使用的情况。若我们在每个规则包反复去编写相同的规则，不仅会大大的增加开发的工作量，大量的编写还可能会出现错误。在这里规则引擎提供了“决策树”的概念：在“决策树”中可以引用其他规则包的规则单位。被引用的规则可以是其他规则包的规则，也可以是其他工程的规则。我们可以在“决策树”中，设置引用规则执行的顺序，也可以设置引用规则执行的条件。最后，“决策树”除了可以引用规则包的规则外，还可以在该决策池中添加“规则”，“规则集”，“决策表”等功能。

2. 基础规则

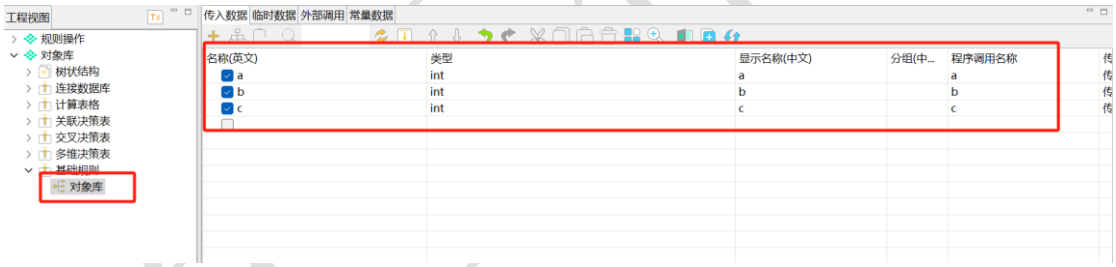
(1) 创建规则包

在“规则操作”下新建规则包“基础规则”：



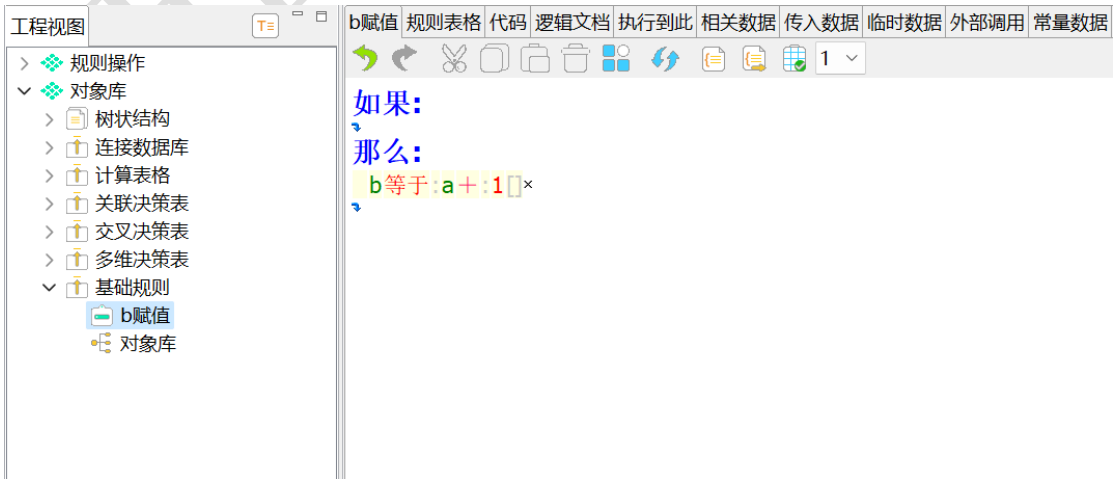
(2) 定义变量

打开“基础规则”规则包，在对象库的传入数据中添加如下变量：

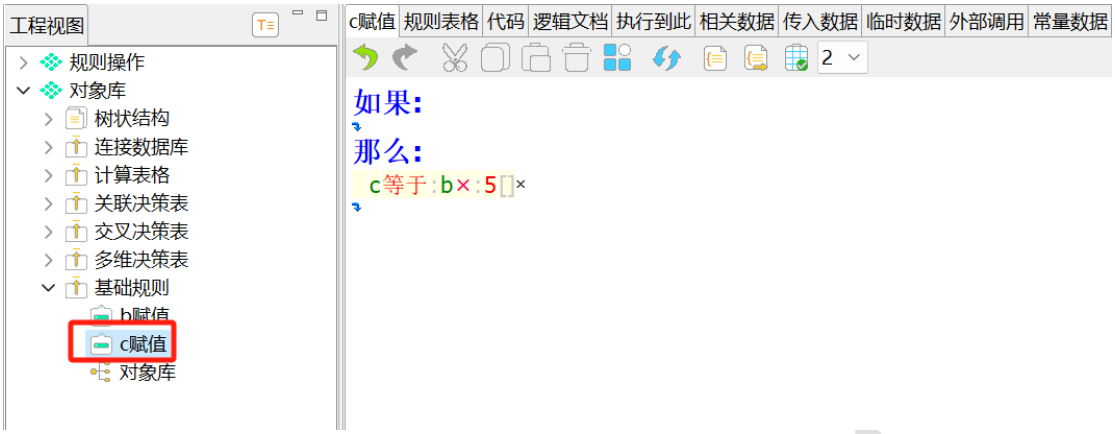


(3) 编写规则

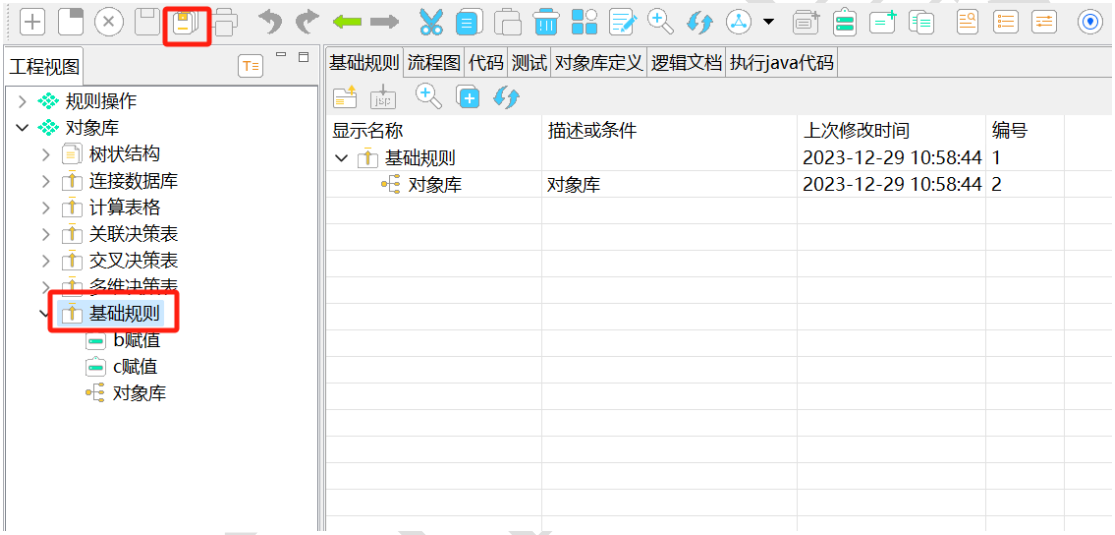
在规则包“基础规则”下添加规则，并命名为“b 赋值”，在编辑窗口中添加动作如下：



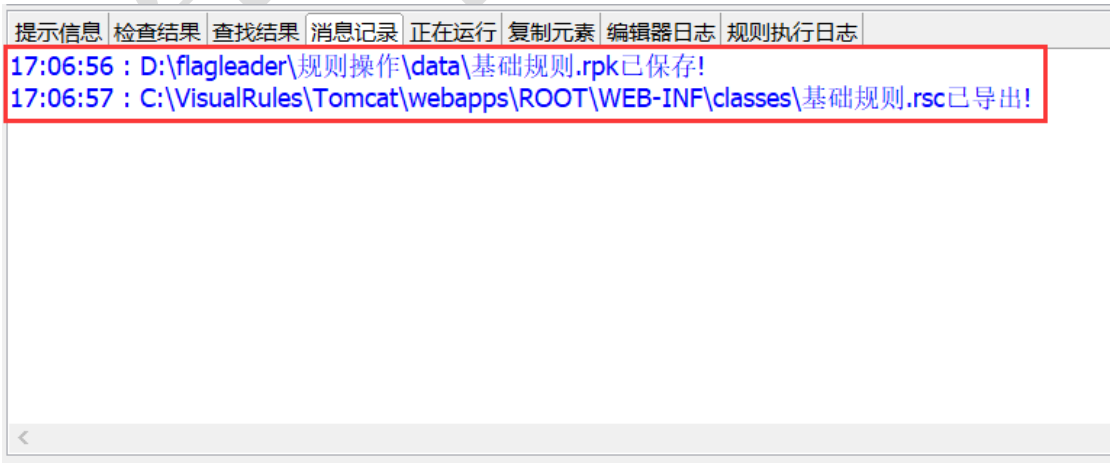
再次添加规则，命名为“c 赋值”，并在编辑窗口中添加如下动作：



“c”赋值完成之后，规则包“基础规则”就编写完成了。点击“全部保存”，这样规则包就会自动保存并编译，如下图所示：



保存并编译后，消息窗口中提示如下：

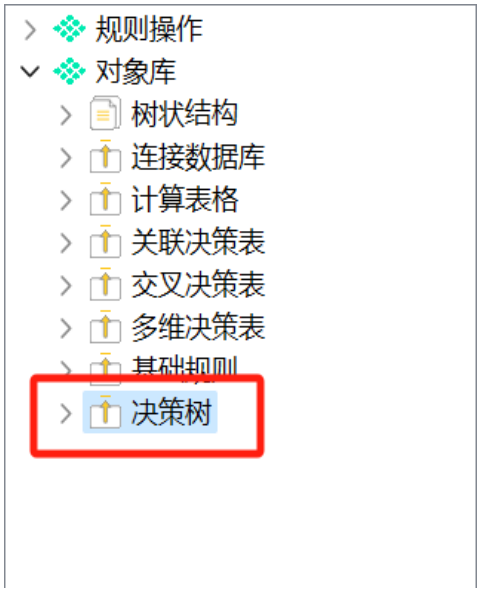


3．决策树

基础规则完成之后，我们需要再创建一个名为“决策树”的规则包。该规则包通过“决策树”的方式对“基础规则”中的规则进行引用。

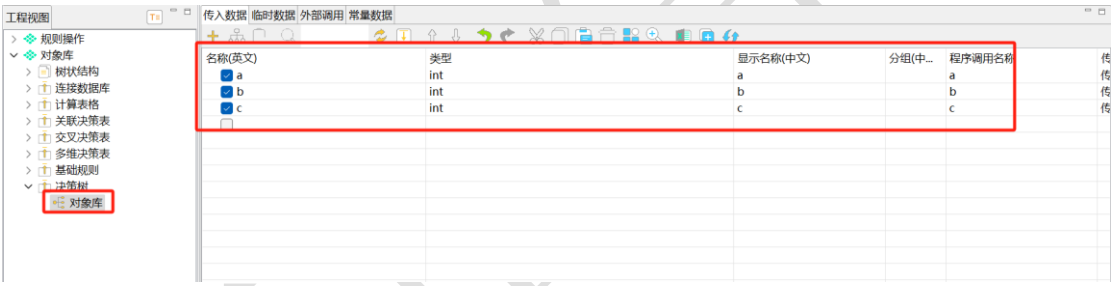
(1) 创建规则包

右键工程“规则操作”，选择“新建规则包”，并命名为“决策树”：



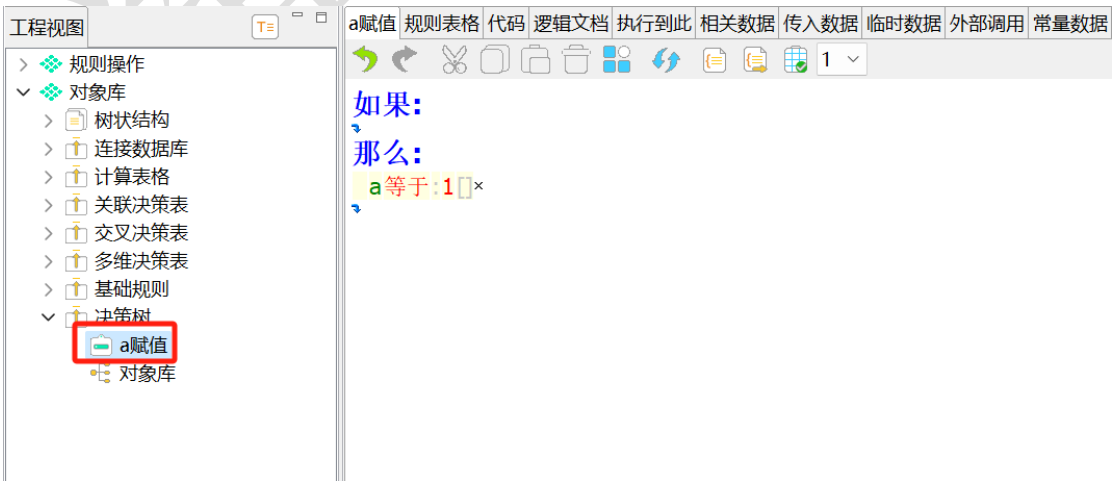
(2) 定义变量

打开规则包，在对象库传入数据中定义如下变量：

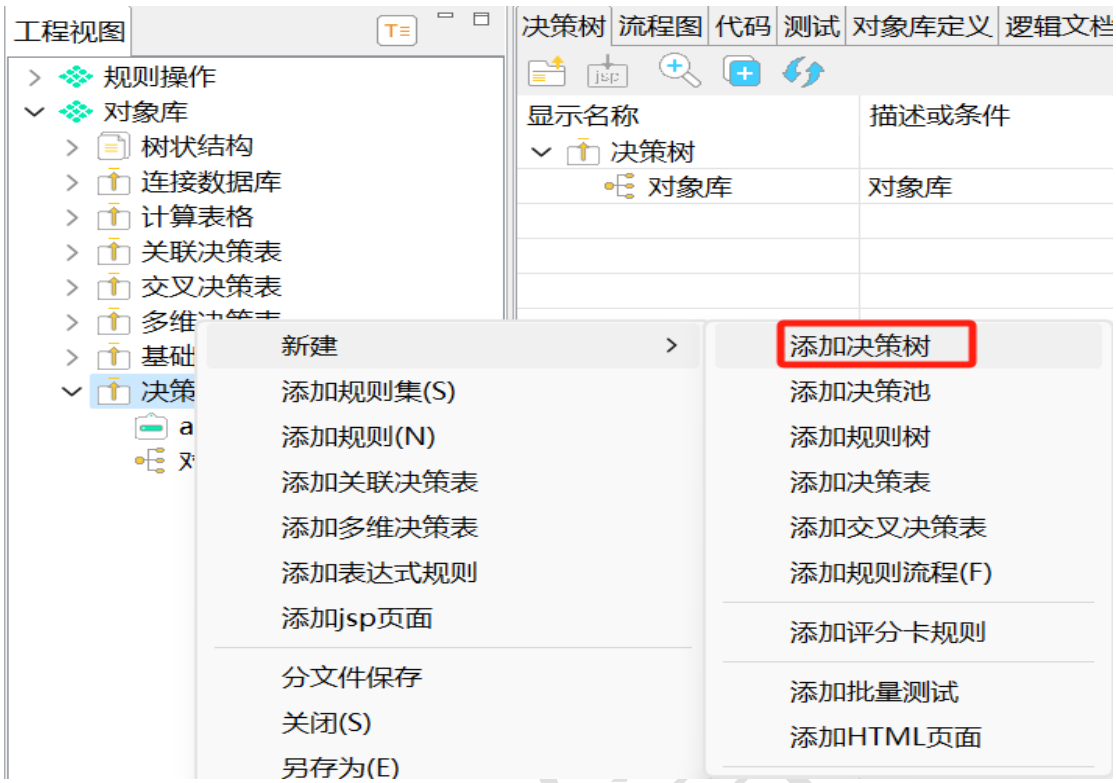


(3) 规则实现

变量定义完成之后，在规则包下添加规则，命名为“a 赋值”，并在编辑窗口中添加动作如下：



规则“a 赋值”逻辑完成之后，我们要添加个“决策树”，添加过程如下图所示：

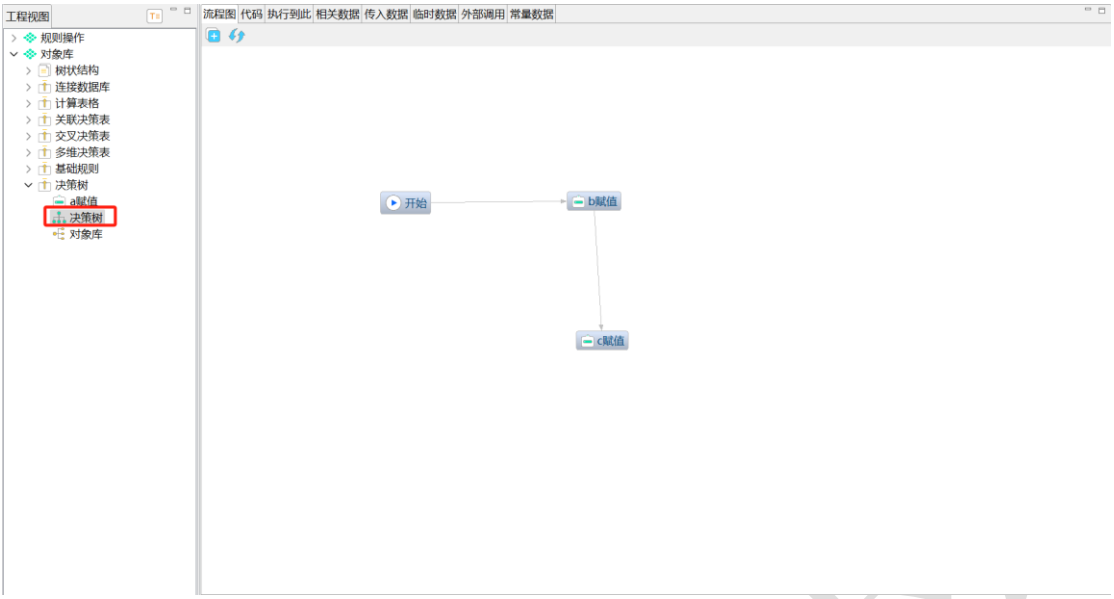


“决策树”添加完成之后，就要在该“决策树”中复用“基础规则”中的规则。

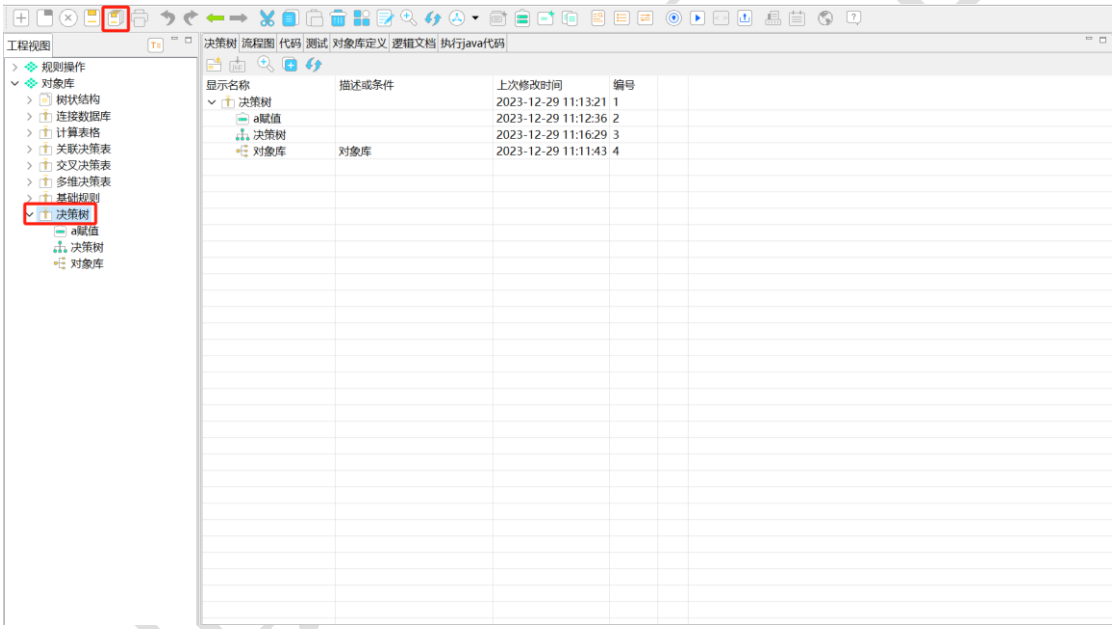
第一步，需要在“决策树”的属性窗口中找到“基础规则”中的规则，然后将规则拖动到“决策树”的编辑窗口“流程图”选项卡中，其步骤如下图所示：

拖动到决策树中之后，流程图窗口如下图所示：

第二步，我们要在“决策树”中，设置规则执行的顺序。如下图所示：首先是从决策树中“开始”开始执行，然后是“b 赋值”，最后再执行“c 赋值”。在设置规则执行顺序中，需按住“Alt”键同时鼠标左键点击拖动进行连接。连线完成之后，“决策树”如下图所示：



连线完成之后，规则包编写完成。点击“  ”保存并编译规则包。

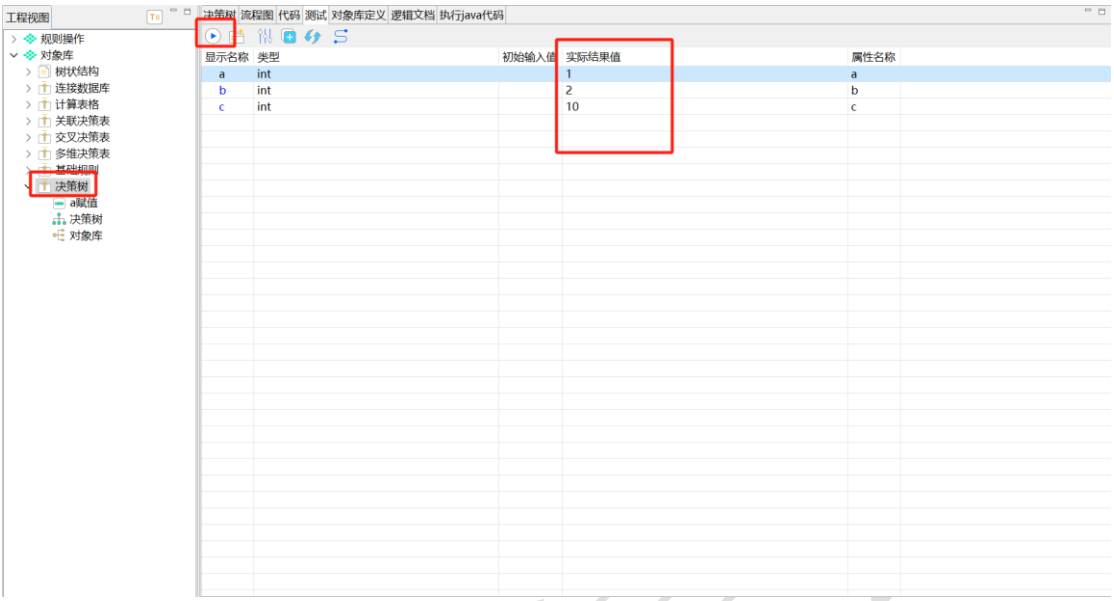


在规则包编译完成之后，在消息窗口中会出现如下所示的提示信息：

这两条记录中，“决策表.rpk”文件是该规则包的工程文件，该文件可直接通过“规则配置器”来打开。“决策树.rsc”文件是该规则包生成的二进制文件。

4．测试

规则包保存、编译完成之后，就可以测试规则包的输出结果。测试过程如下所示：



3.6. 决策池

1．功能介绍

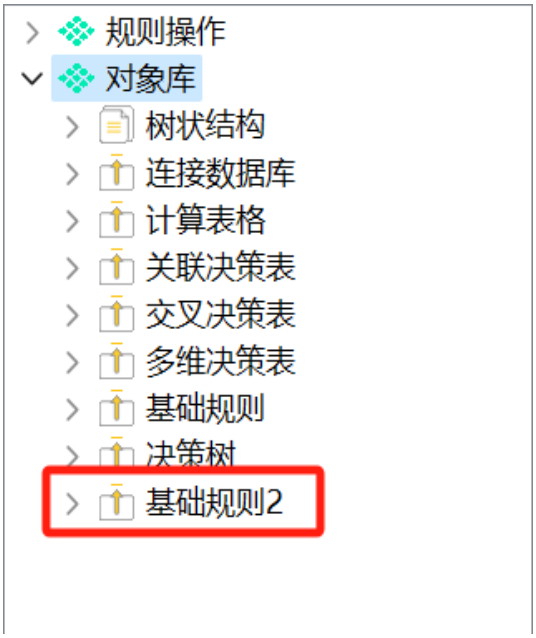
在规则开发过程中，往往会遇到某个规则会反复被使用的情况。因此，我们总是希望能通过某种方式调用这些规则，也就是所能实现规则复用。我们知道在“决策树”中，不仅可以实现规则的复用，而且还可设置复用规则执行的条件和复用规则间的条件。但是在“决策池”中，会执行“决策池”里所有的被复用的规则，被复用规则的执行的顺序与规则被引用的先后顺序是一致的。

2．基础规则

首先需要创建一个名为“基础规则”的规则包，该规则包下的规则，作为“决策池”的复用规则。

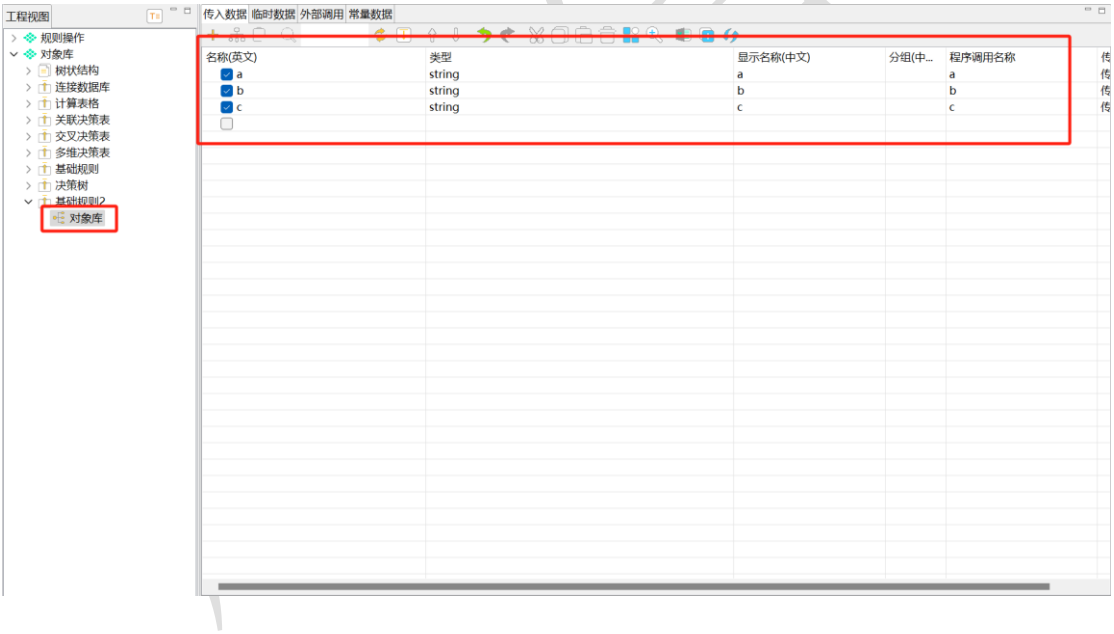
(1) 规则包创建

右键点击工程“规则操作”，点击“新建规则包”，创建一个名为“基础规则 2”的规则包。如下图所示：

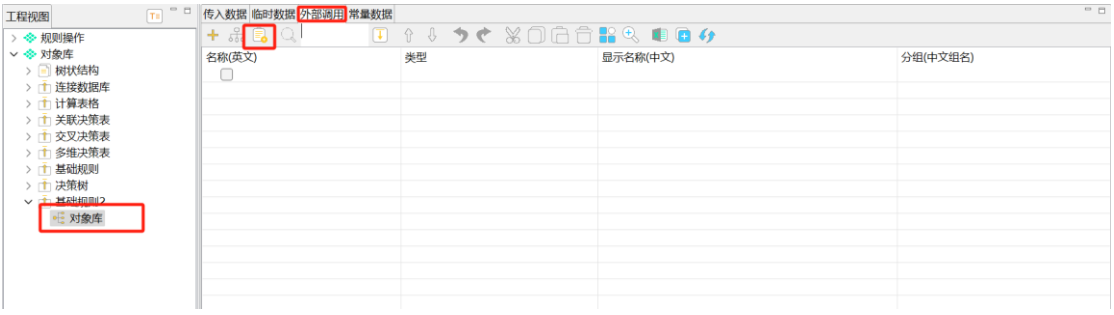


(2) 变量定义

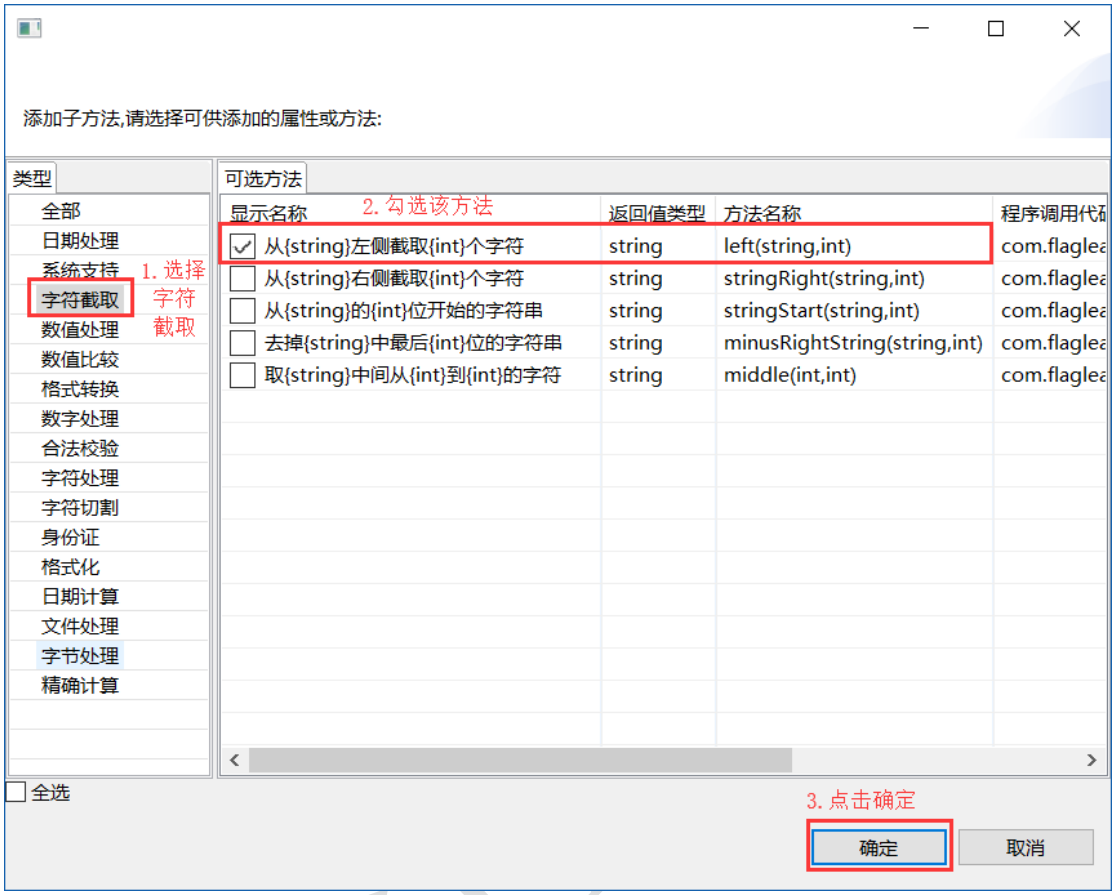
打开该规则包，在对象库传入数据中，需要定义三个“string”类型变量：a（a），b（b），c（c）。如下图所示：



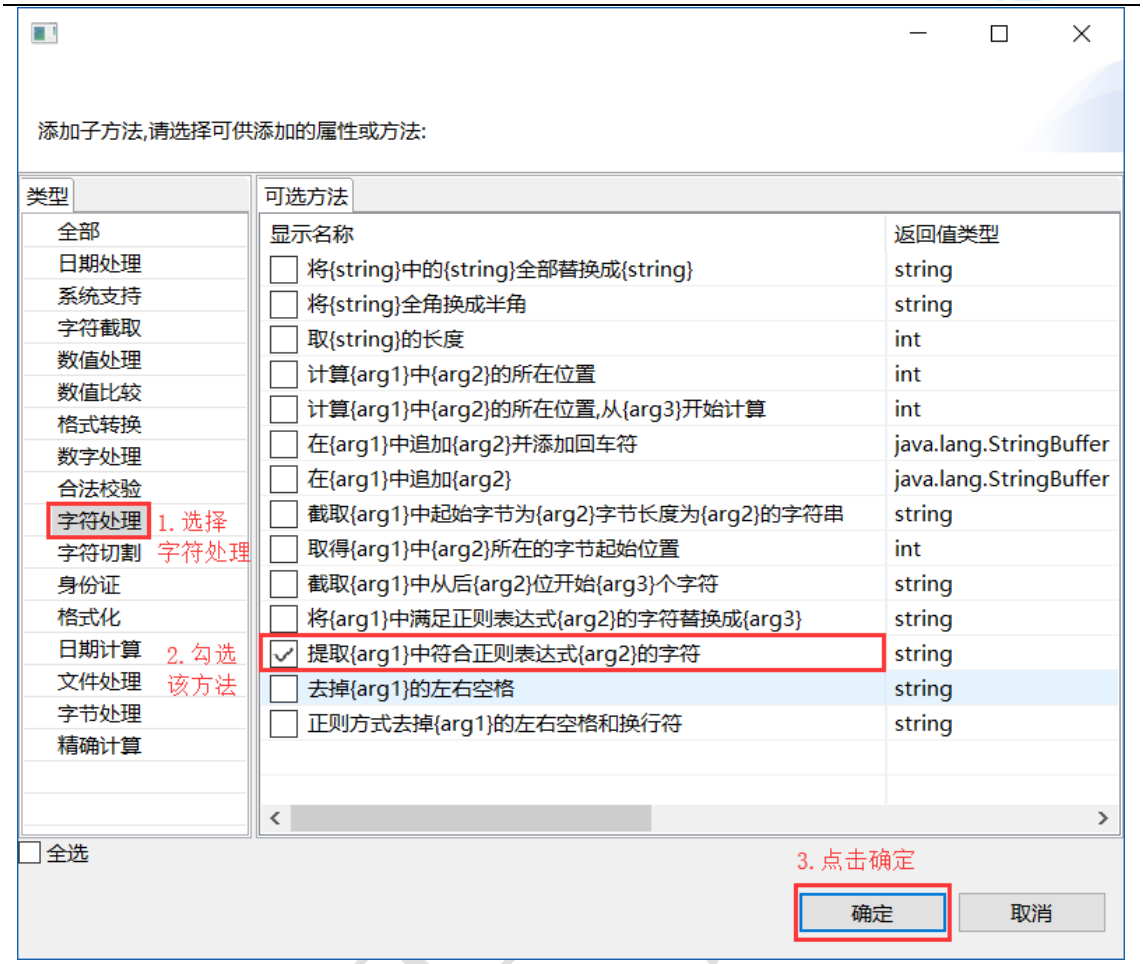
在对象库的外部调用中，点击“添加公式”按钮添加两个方法。操作步骤如下图所示：



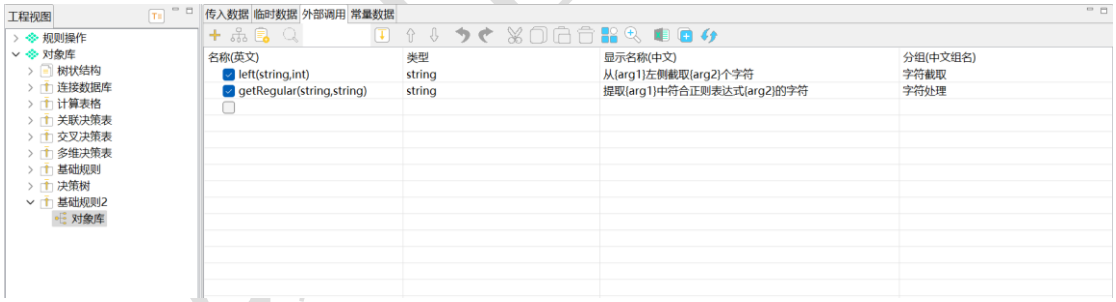
这些方法是规则引擎封装的一些常用方法，可以在“外部调用”中添加这些常用方法，然后在规则中去调用。



重复上述步骤，添加第二个方法：

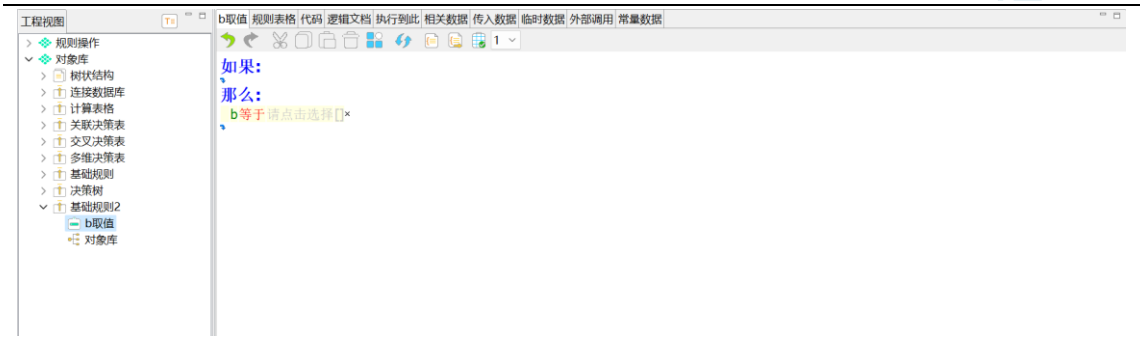


在对象库外部调用中查看刚刚添加的两个方法：

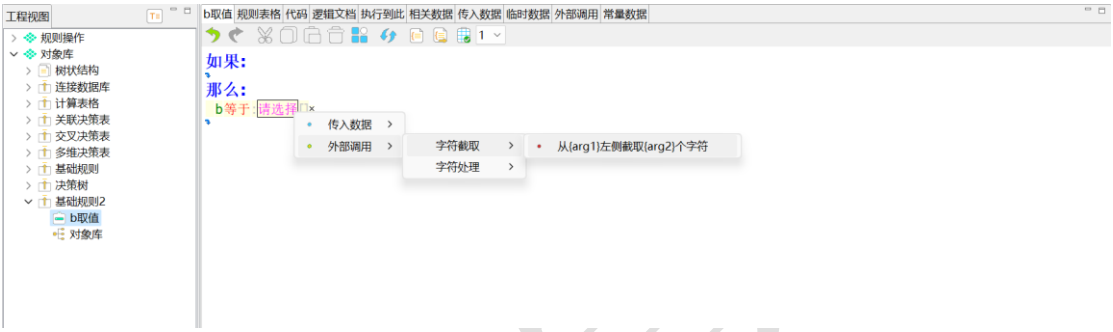


(3) 规则编写

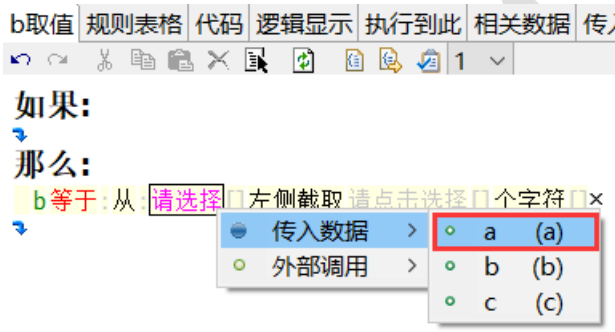
变量定义完成之后，要在规则包中实现具体规则逻辑。右键选择规则包“基础规则 2”添加名为“b 取值”的规则，b 取值并在编辑窗口中添加动作：

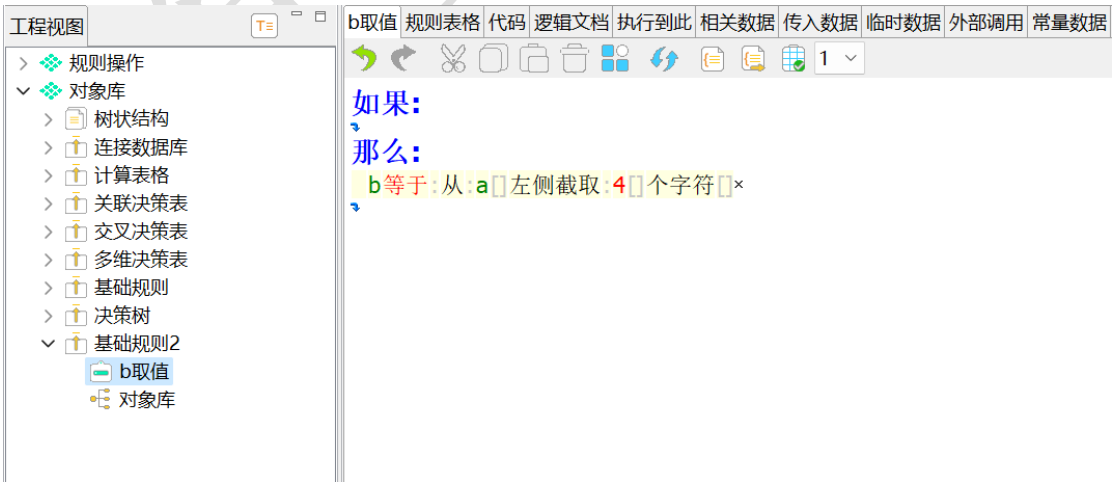


对变量 b 进行赋值，选择选择值→外部调用→字符截取，操作如下：

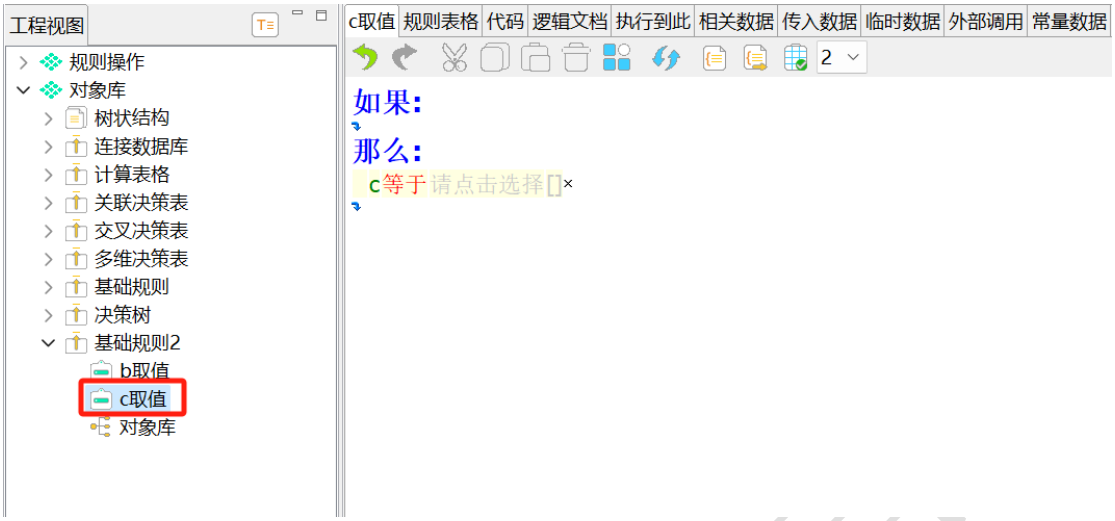


对 arg1 和 arg2 赋值，其中 arg1 的值为选择值→传入数据→a；arg2 的值为输入常量数据→输入 4。

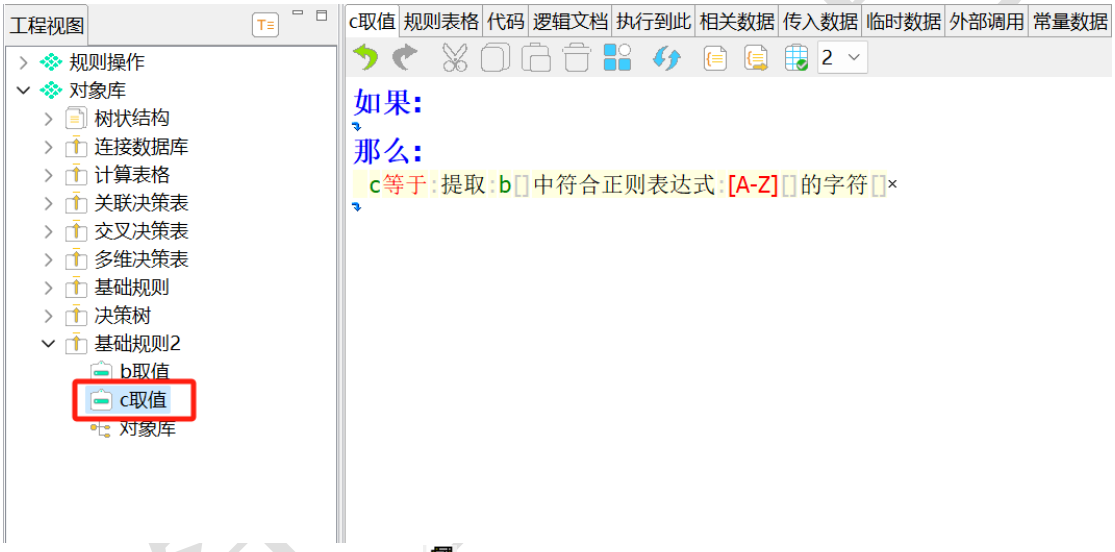




完成之后，再在规则包“基础规则 2”下添加规则“c 取值”，并添加动作如下：



对 c 进行赋值，方法步骤同上：



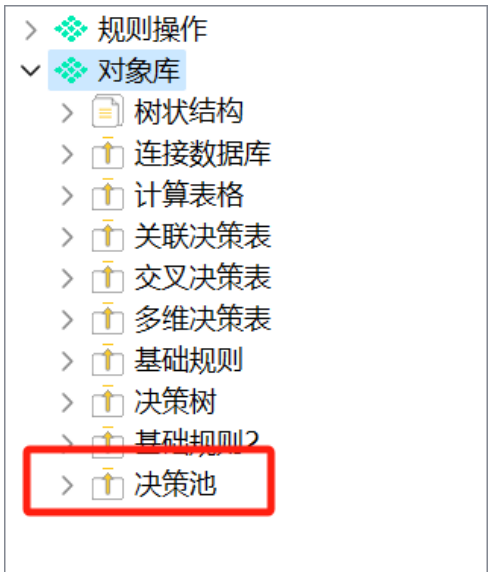
完成之后，点击工具栏中的“ ”保存并编译。

3. 决策池

规则包“基础规则 2”完成之后，我们再创建一个名为“决策池”的规则包。在该规则包的“决策池”中，来调用“基础规则 2”中的规则。

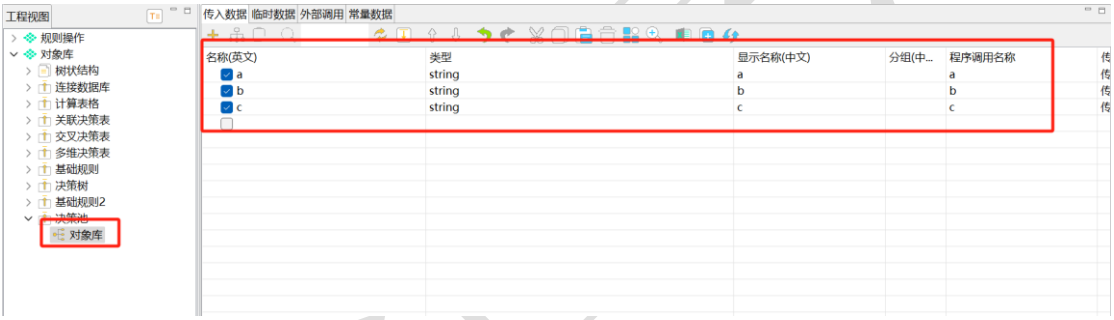
(1) 规则包创建

右键点击“规则操作”的工程，选择“新建规则包”，添加一个名为“决策池”的规则包。如下图所示：



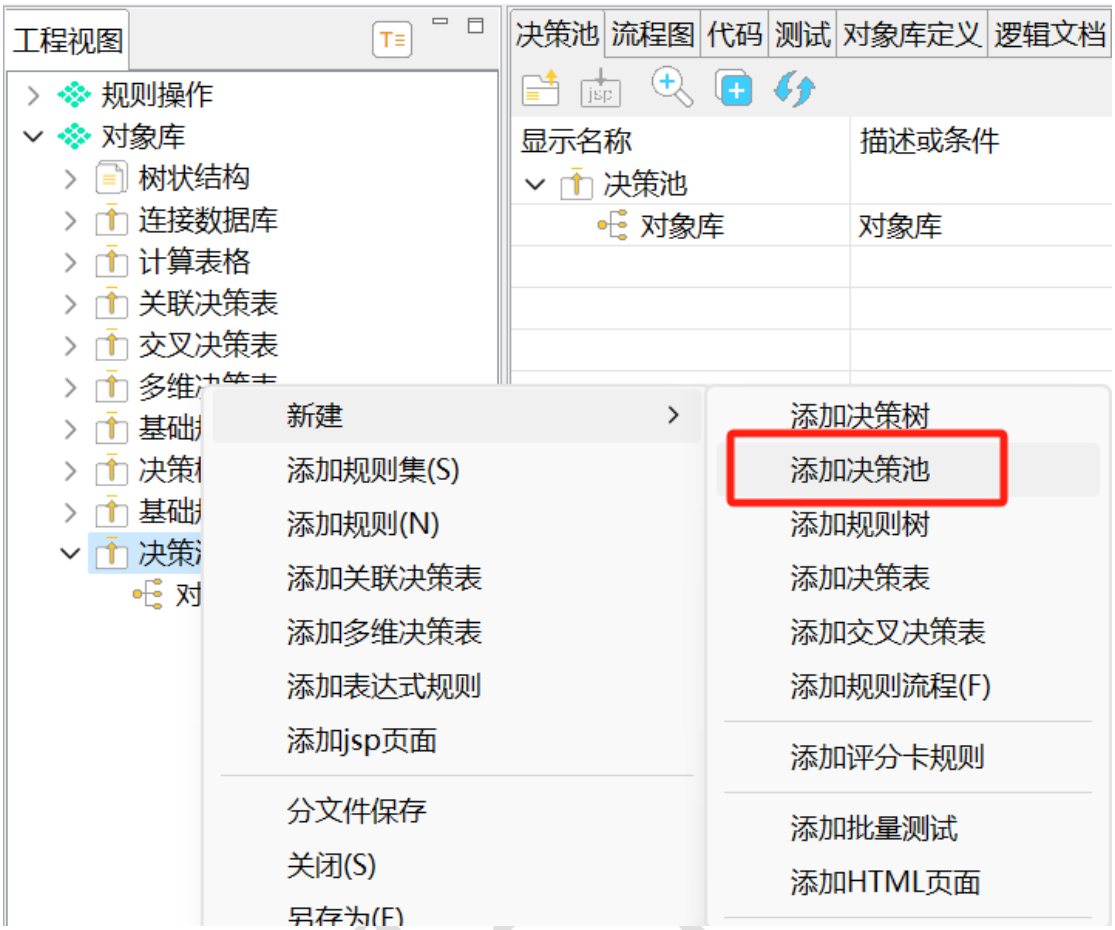
(2) 变量定义

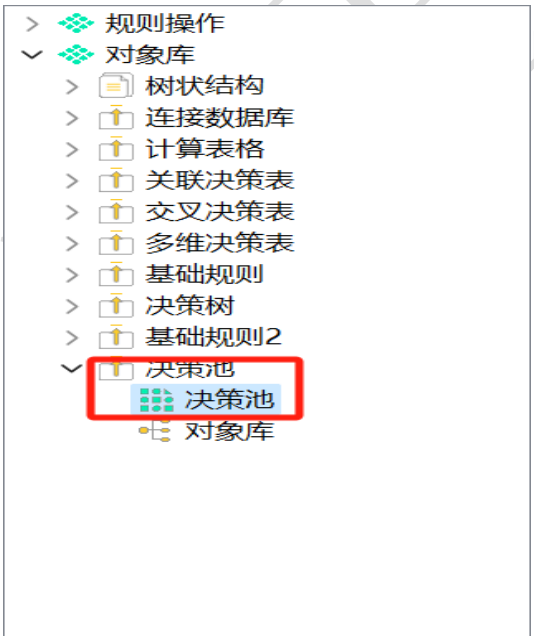
与“基础规则 2”中一样，在“决策池”规则包中，我们也需要定义三个“String”类型的变量“a”、“b”、“c”。如下图所示：



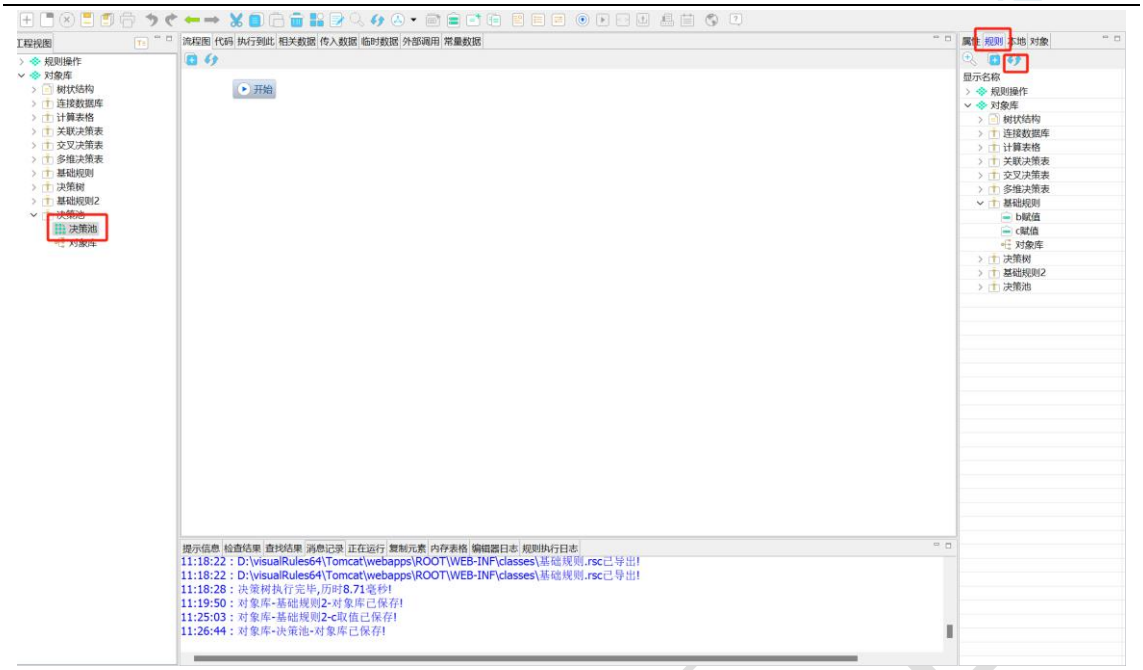
(3) 规则实现

在规则包中添加“决策池”，在“决策池”中实现对“基础规则 2”中的规则的复用。添加“决策池”过程如下图所示：



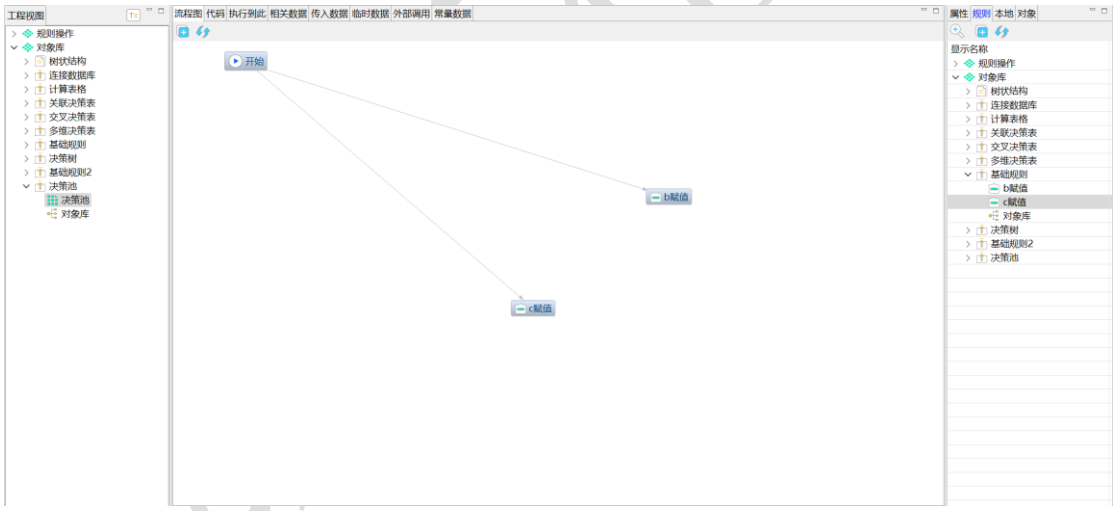


如果要实现对“基础规则 2”中的规则复用，首先要在“决策池”的属性栏的“规则”中找到该规则包，如下所示：



选择规则包“基础规则 2”下的规则，将其拖动到“决策池”的“流程图”中。注意拖动顺序，先拖入的规则在“决策池”中先执行。

拖入完成之后，如下图所示：



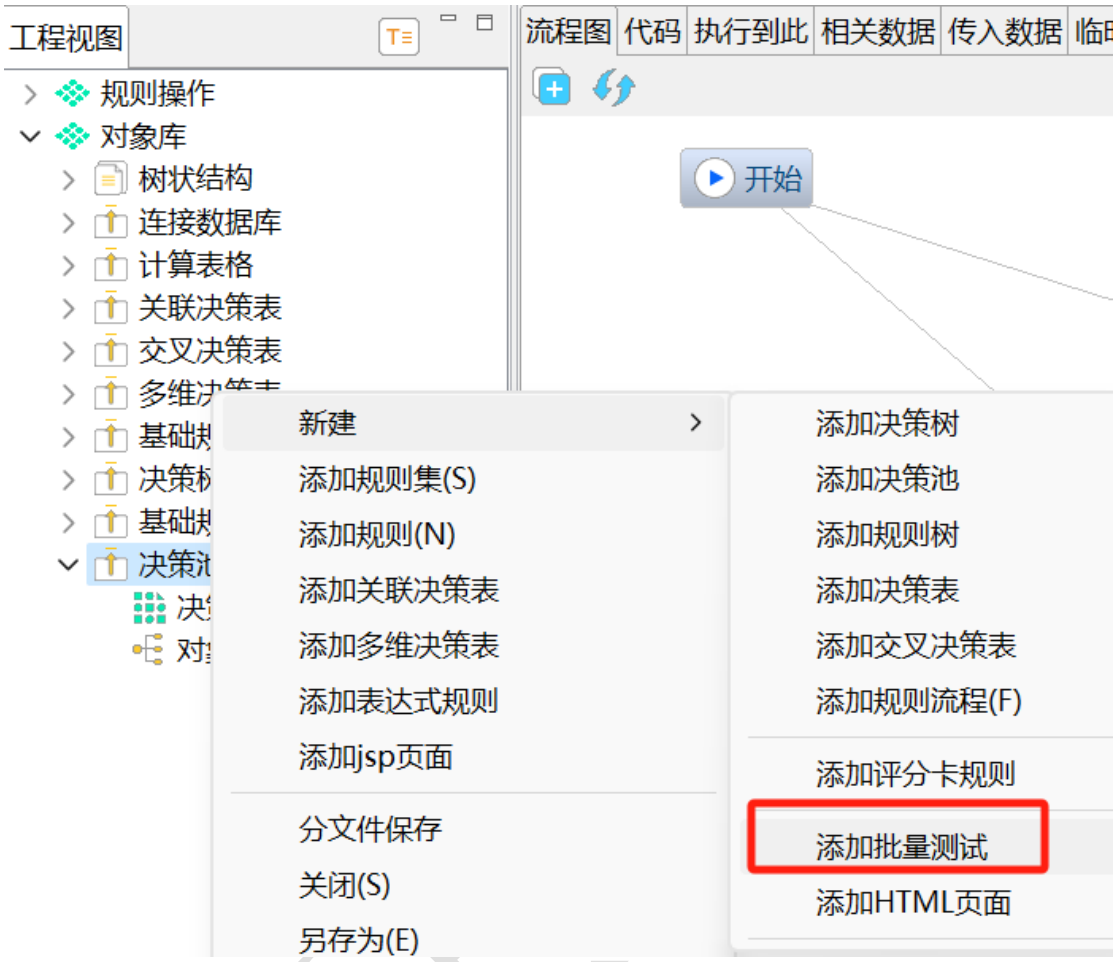
这样“决策池”就实现了对规则“b 取值”、“c 取值”复用：规则包“决策池”将会依次执行“b 取值”、“c 取值”中的规则逻辑。

4．批量测试

我们在规则逻辑完成之后，要测试规则包正确性。给规则规则包添加个“批量测试”，然后在“批量测试”中添加若干测试用例、期望结果值，最后我们只需比对测试用例的期望结果值与规则包的实际输出结果值是否相同。若相同，则测试完成，规则包未发现错误；若不同，测试不通过。

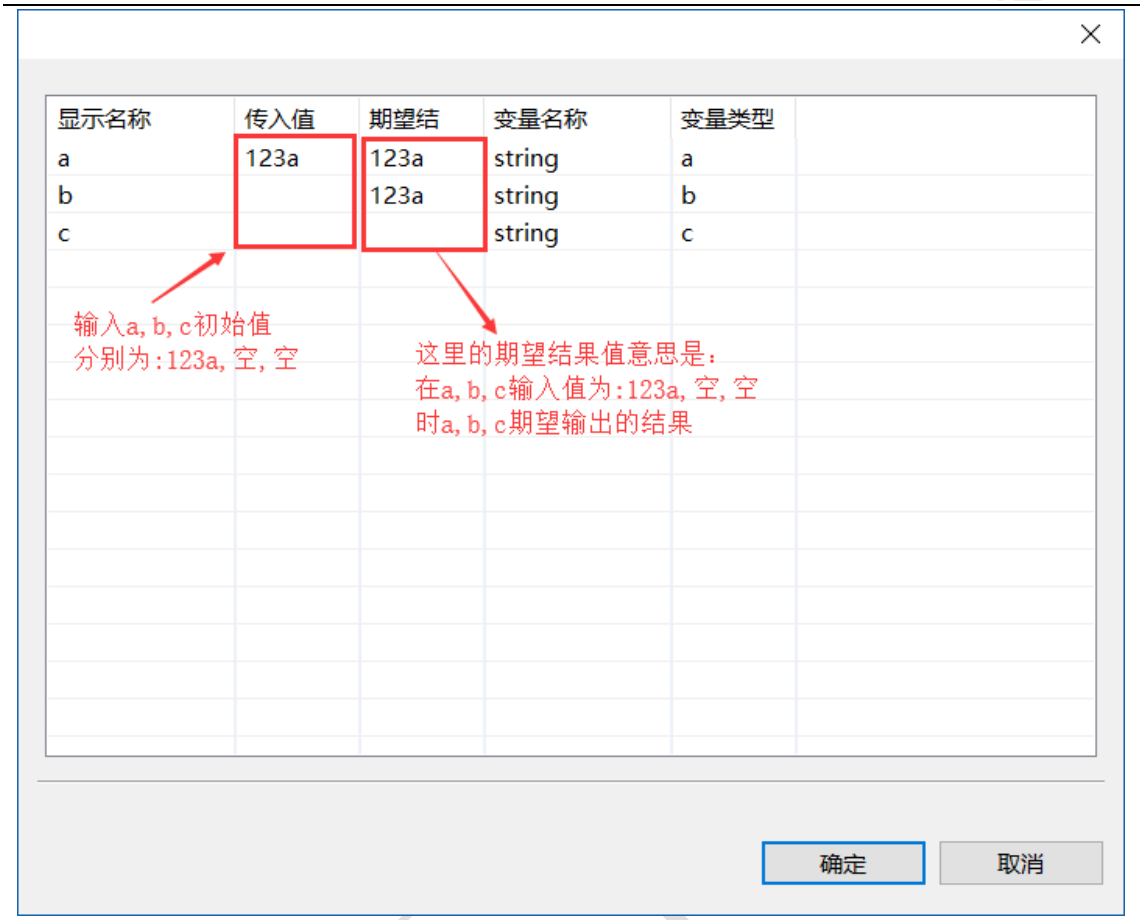
（1）添加批量测试

右键规则包，选择新建→添加批量测试：

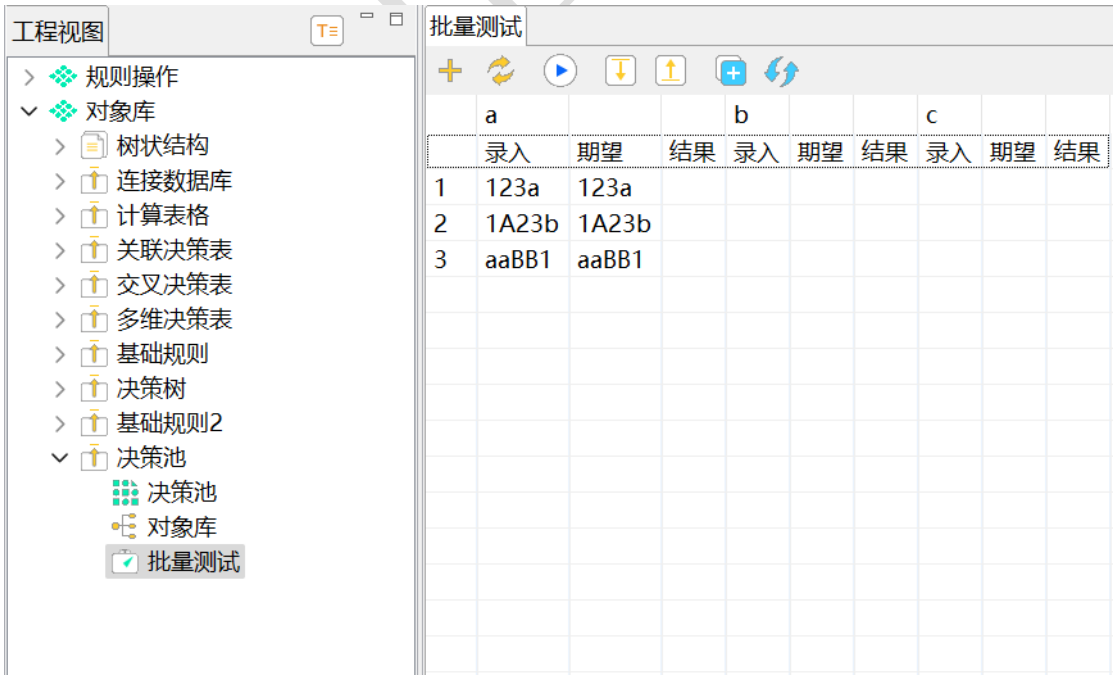


(2) 添加测试用例

“批量测试”添加完成之后，我们要在该“批量测试”下添加测试用例。点击编辑窗口工具栏中的“”，在弹出框中添加测试用例：



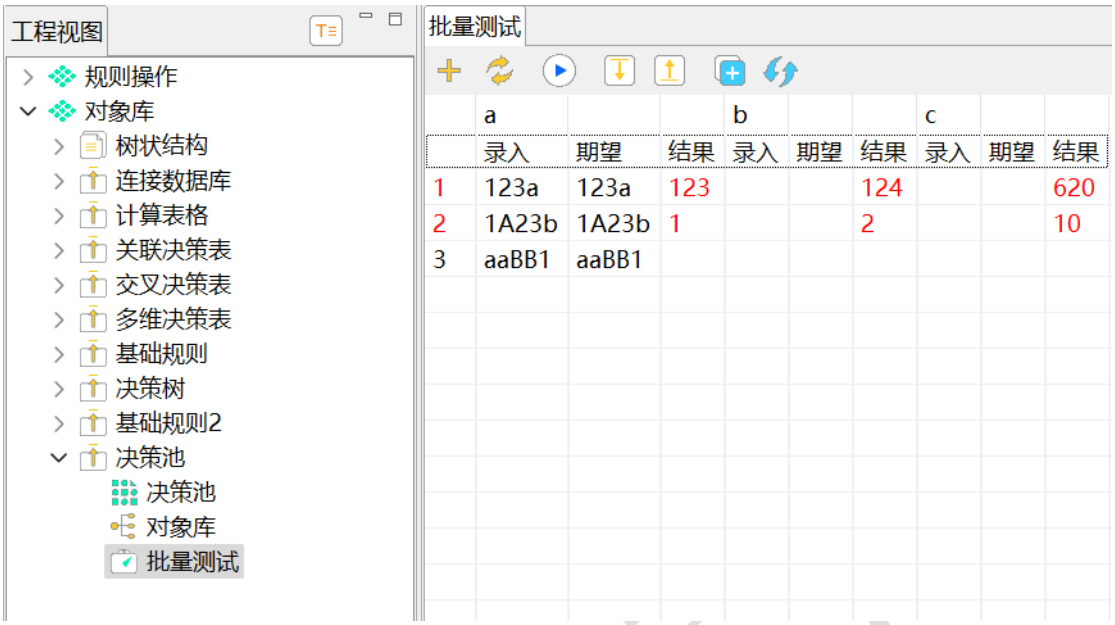
按照上述步骤，我们再给“批量测试”添加两个测试用例。添加完成后，“批量测试”如下图所示：



测试用例添加完成之后，点击“”保存并编译规则包。

(3) 测试结果

最后测试该规则包，以下是在这些测试数据下实际输出的结果值。测试过程如下所示：



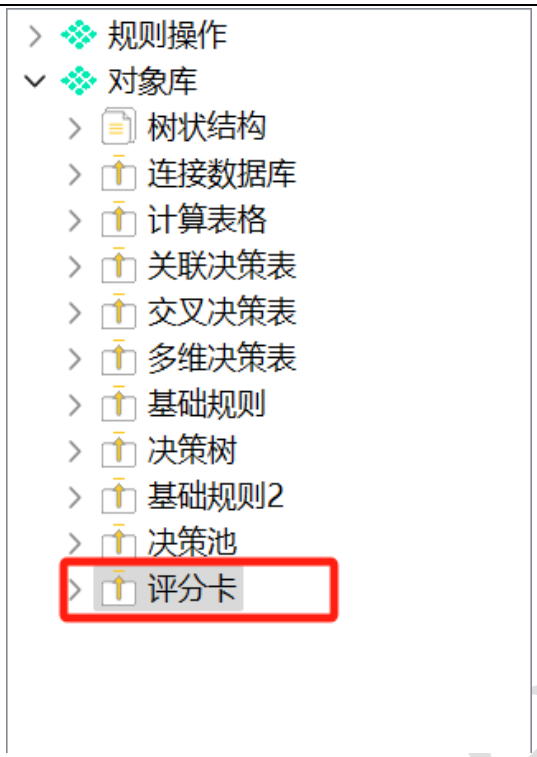
从上图中可以发现，所有的期望结果与实际输出结果相同，规则包测试完成！

3.7. 评分卡

1. 创建规则包

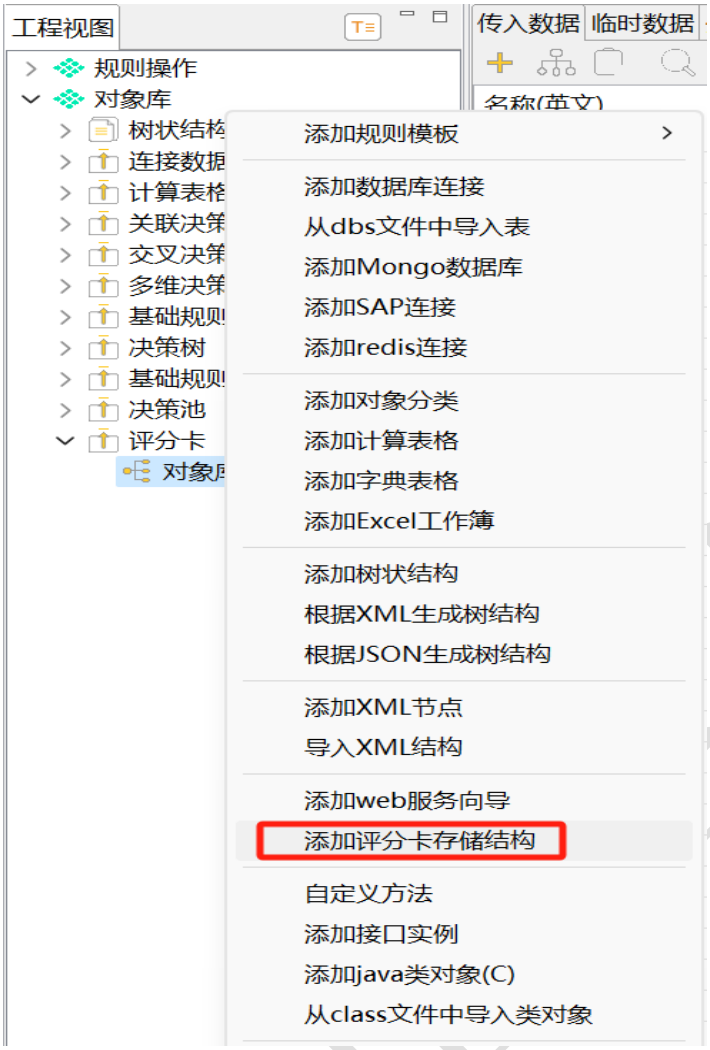
在“规则操作”工程下创建规则包“评分卡”。

右键点击“规则操作”工程，选择新建规则包，并将其命名为“评分卡”：

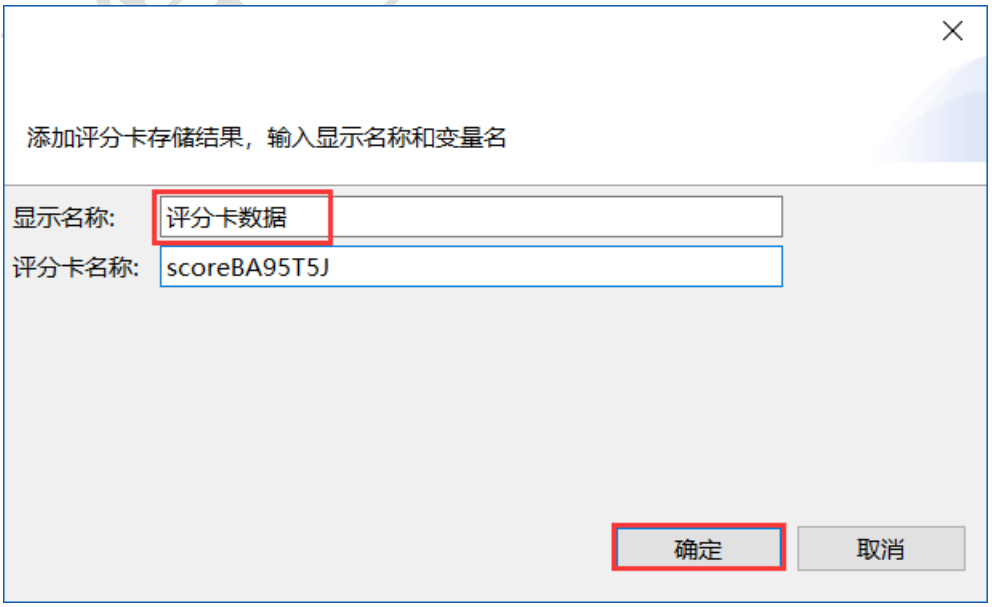


2．添加评分卡存储结构

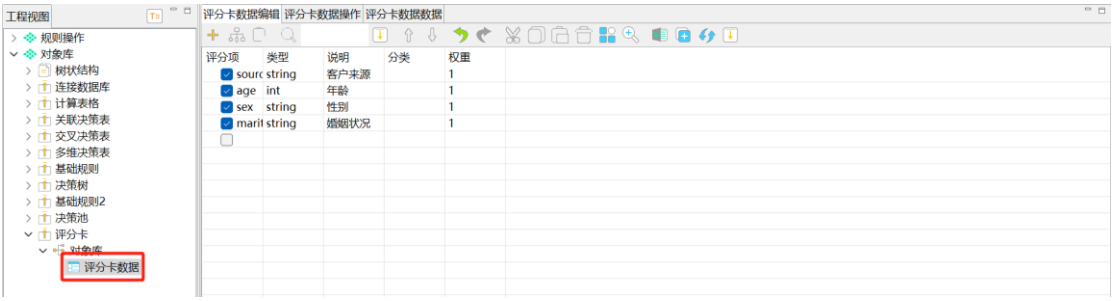
打开“评分卡”规则包，右键对象库选择“添加评分卡存储结构”：



在弹出框中对“评分卡存储结构”进行命名，完成后点击确定：

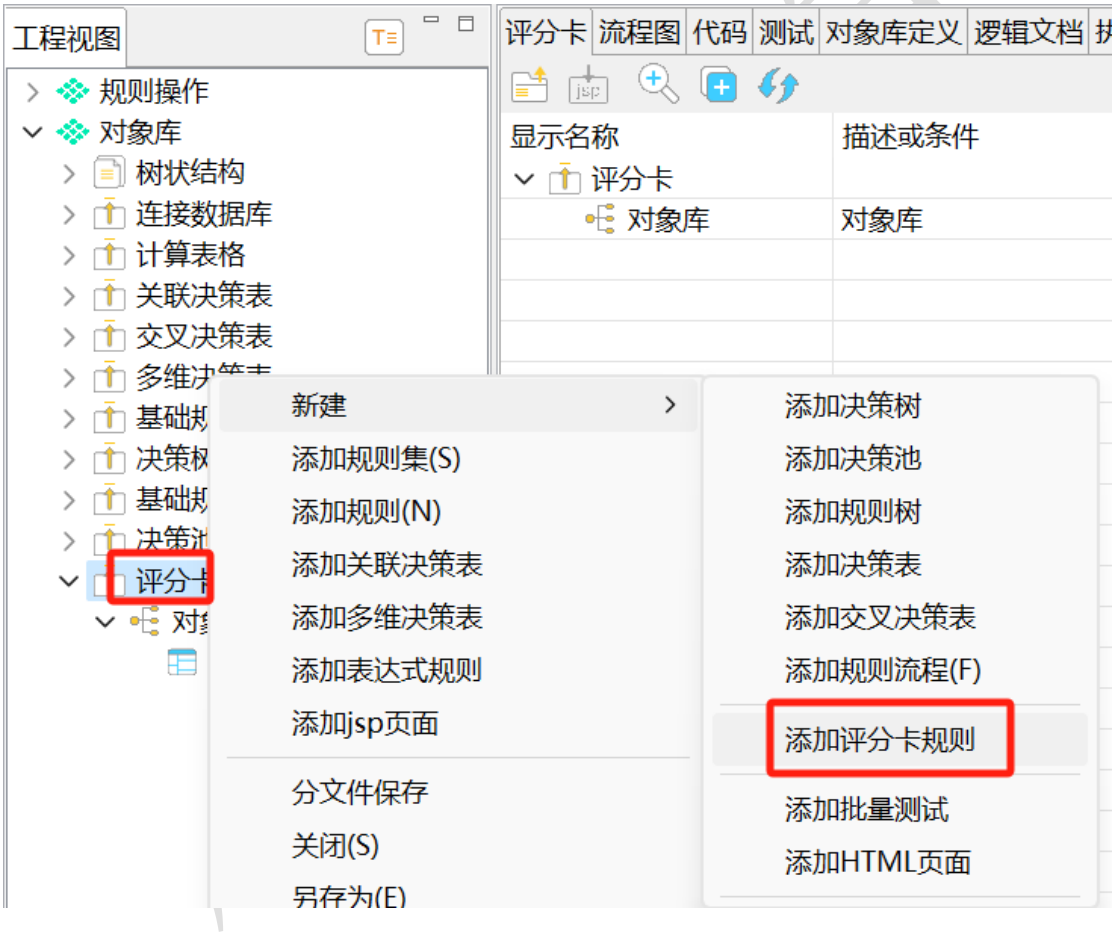


点击“评分卡数据”，在编辑中添加变量“客户来源（source）、年龄（age）、性别（sex）、婚姻状况（maritalStatus）”，其中年龄为 int 类型，其他的都为 String 类型：



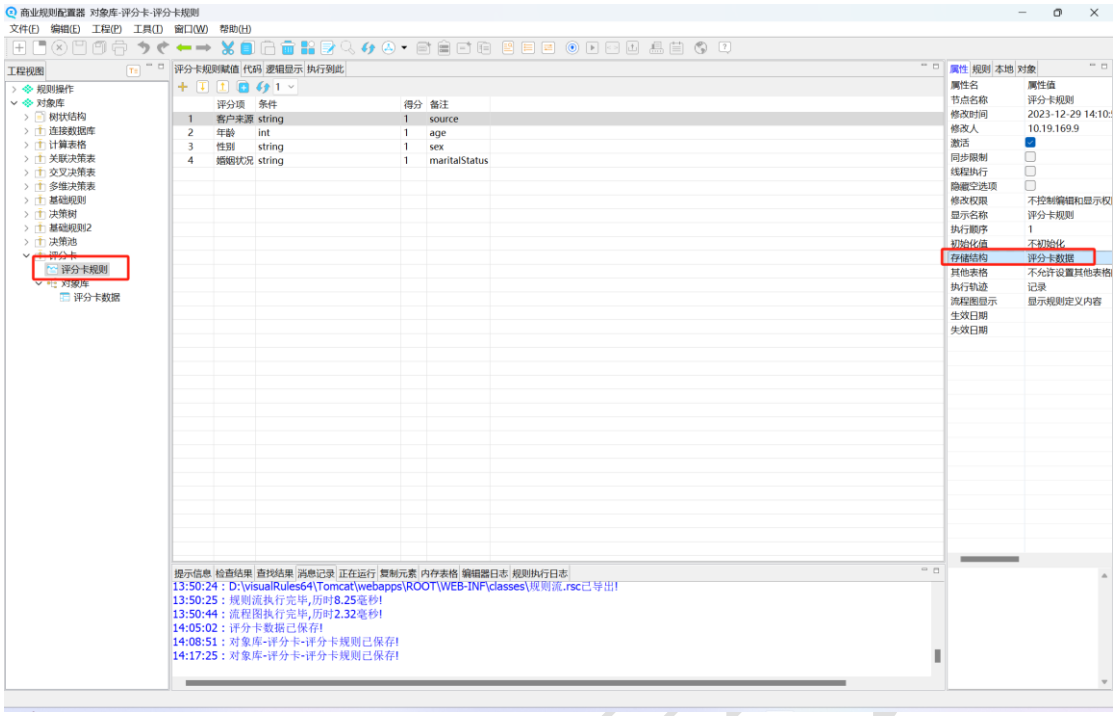
3．添加评分卡规则

选择规则包“评分卡”，右键选择 新建→添加评分卡规则：

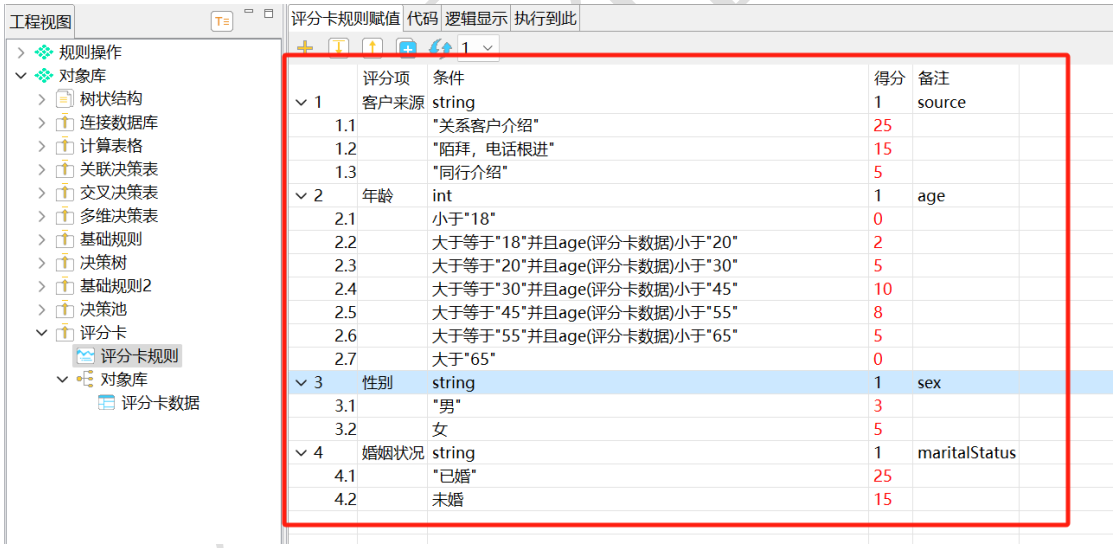


在“评分卡规则”中添加“评分项”。

在属性窗口中，双击属性名“存储结构”所对应的属性值单元格，出现下拉列表，选择“评分卡数据”，这时在中间的编辑窗口中会自动添加评分项：



有了“评分项”之后，我们需要给这些“评分项”添加条件和得分。条件和得分信息如下图所示：

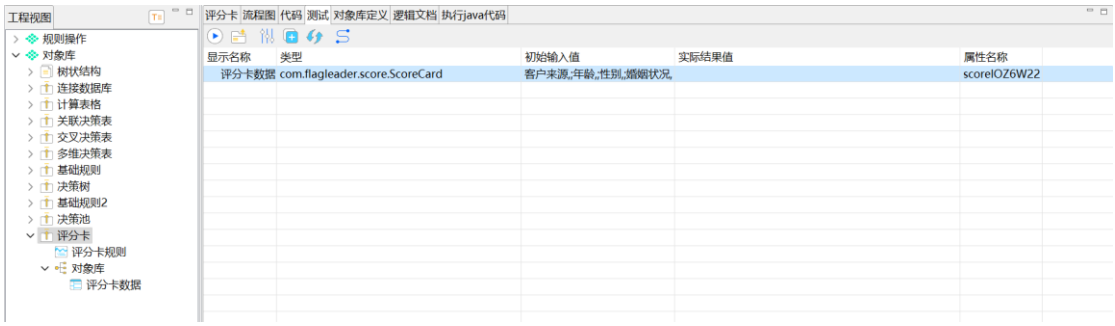


	评分项	条件	得分	备注
1	客户来源	string	1	source
1.1		"关系客户介绍"	25	
1.2		"陌拜，电话根进"	15	
1.3		"同行介绍"	5	
2	年龄	int	1	age
2.1		小于"18"	0	
2.2		大于等于"18"并且age(评分卡数据)小于"20"	2	
2.3		大于等于"20"并且age(评分卡数据)小于"30"	5	
2.4		大于等于"30"并且age(评分卡数据)小于"45"	10	
2.5		大于等于"45"并且age(评分卡数据)小于"55"	8	
2.6		大于等于"55"并且age(评分卡数据)小于"65"	5	
2.7		大于"65"	0	
3	性别	string	1	sex
3.1		"男"	3	
3.2		女	5	
4	婚姻状况	string	1	maritalStatus
4.1		"已婚"	25	
4.2		未婚	15	

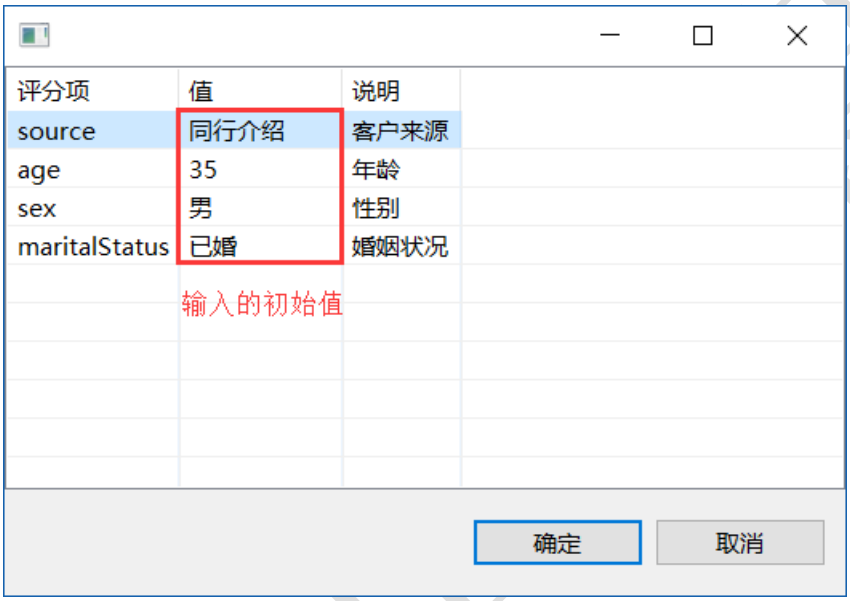
完成之后，点击“  ”保存并编译规则包。

4．测试

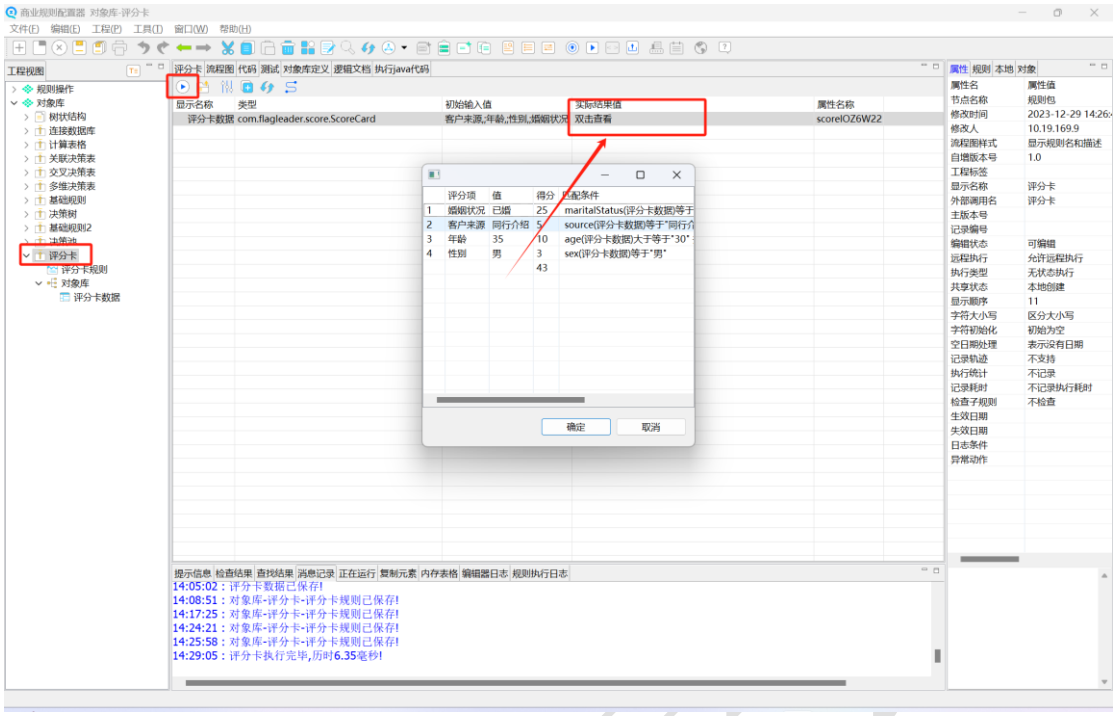
选择“评分卡”规则包，在中间编辑窗体中选择“测试”选项卡，输入初始值进行测试：



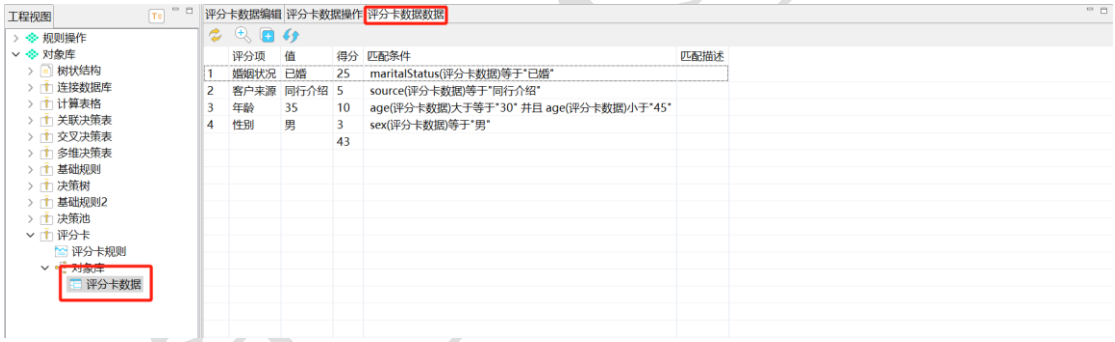
由于这里“评分卡数据”是表格结构，因此要对表中的变量逐个赋值。双击“评分卡数据”对应的“初始输入值”，在弹出窗口中输入初始值：



点击“ ”执行，结果如下：



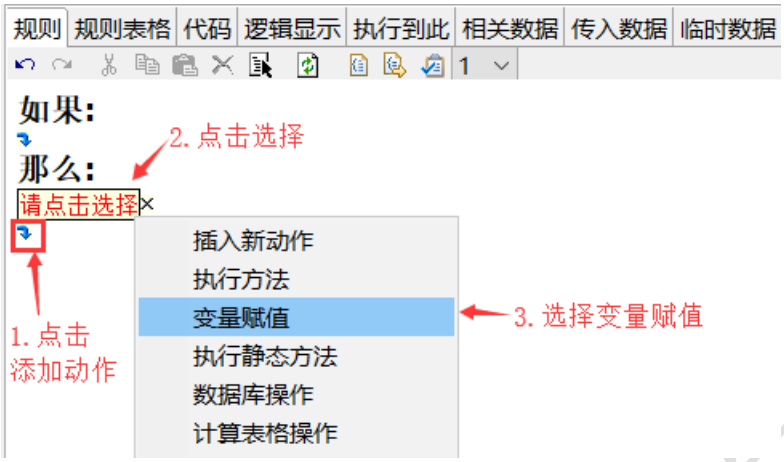
在“评分卡数据”中查看刚刚测试的结果数据，如下图所示：



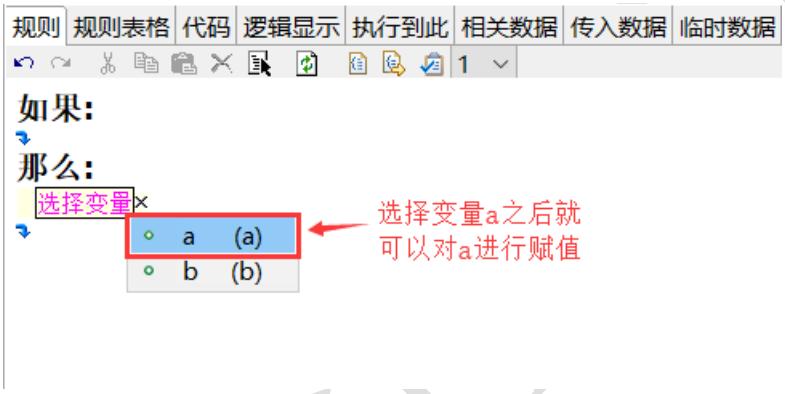
3.8. 表达式规则

在规则配置器上编写规则时，一般都是有操作向导提供我们点击选择。比如我们要选择

给变量“a”赋值，默认的方法如下图所示：



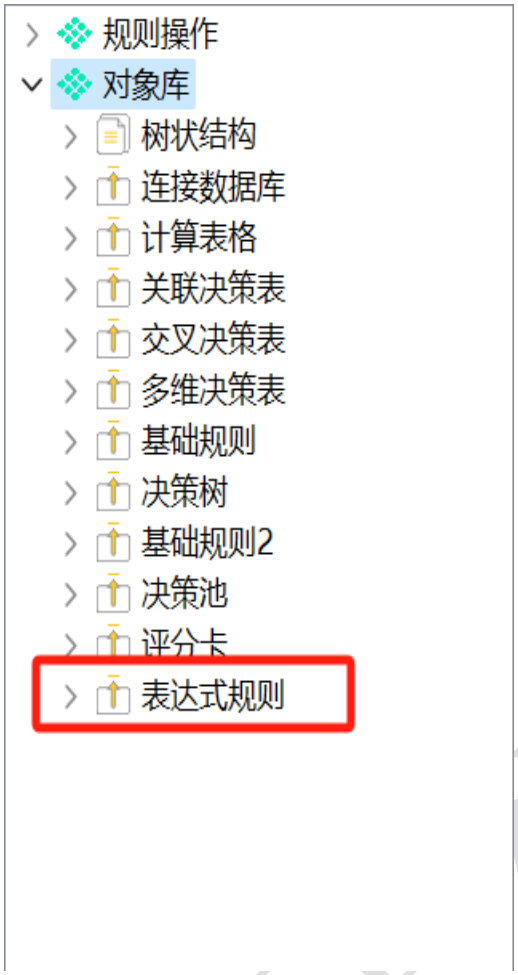
我们可以通过以上三个步骤，选到变量赋值，接下来我们要选择对象库中具体需要定义的变量，如下图所示：



我们可以看到上述的赋值过程是通过鼠标点击选择的，但是我们在表达式规则中，开发人员可以完全通过中文描述来实现。

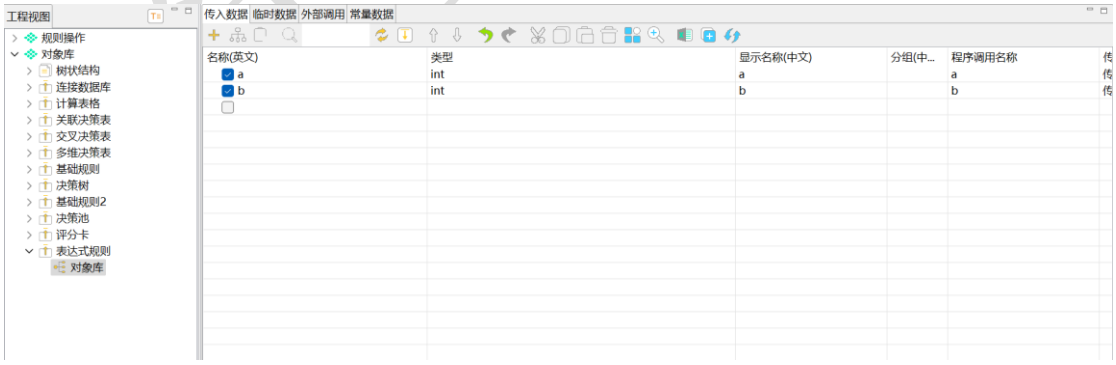
1. 规则包创建

在“规则操作”工程下新建“表达式规则”规则包：



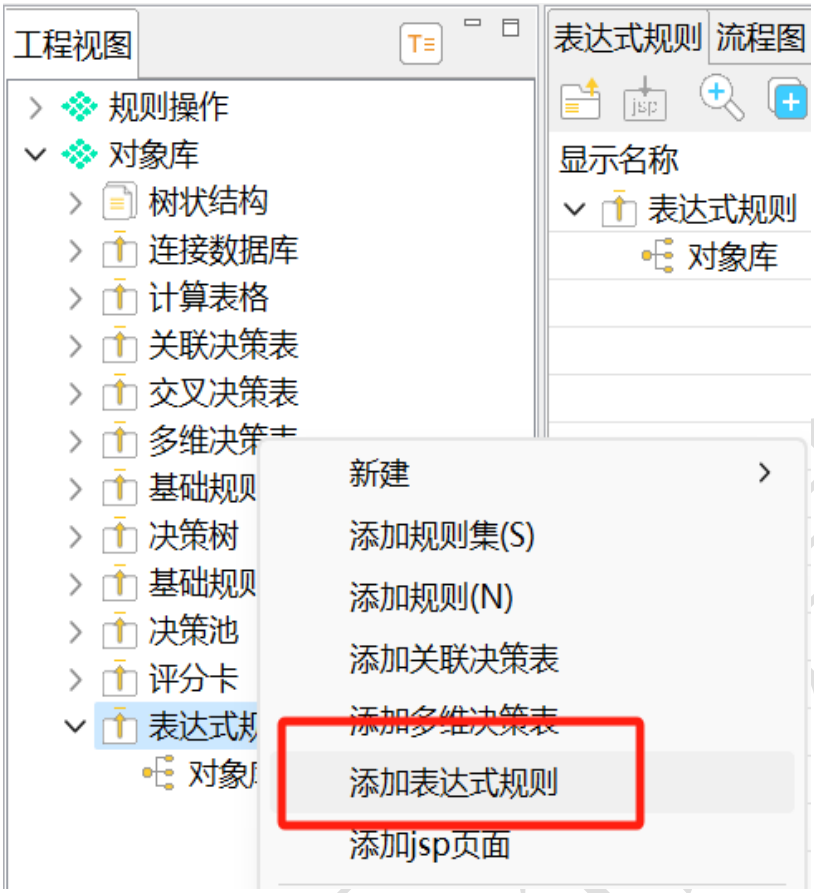
2．定义变量

打开“表达式规则”规则包，在对象库中分别添加变量“a（a）、b（b）”两者均为 int 类型：

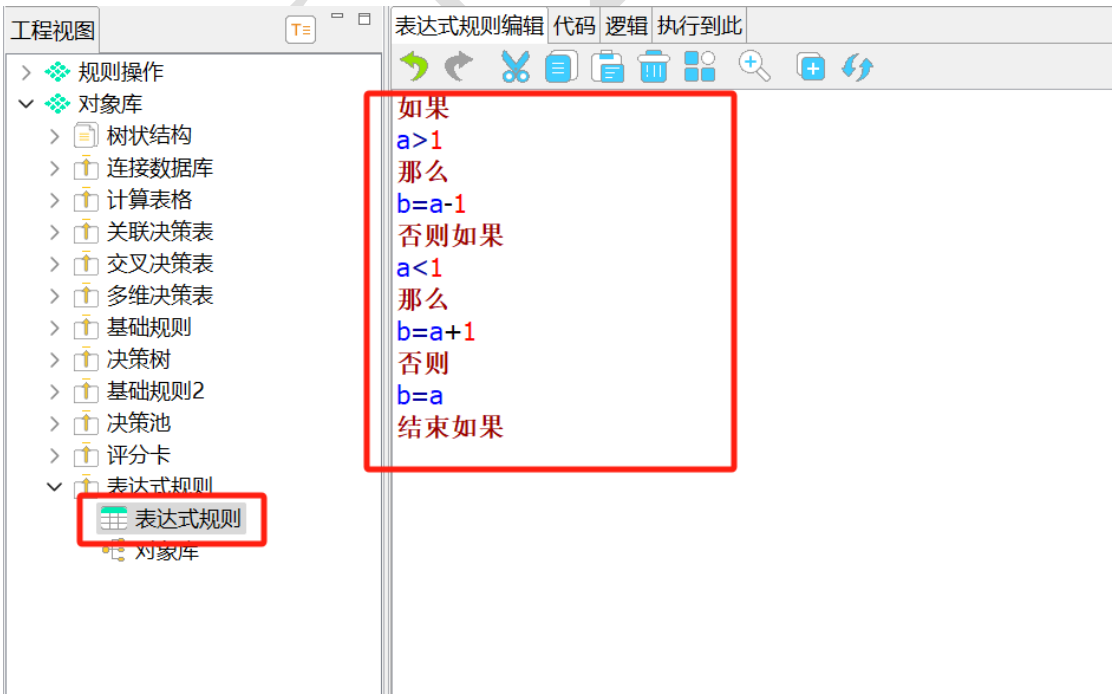


3．添加表达式规则

选择“表达式规则”规则包，右键添加表达式规则：



创建好了“表达式规则”之后，我们需要设置“表达式规则”的逻辑，这里我们只需要直接写中文就行了，完成后的“表达式规则”如下图所示：



（注意：在结束的时候要加上“结束如果”。）

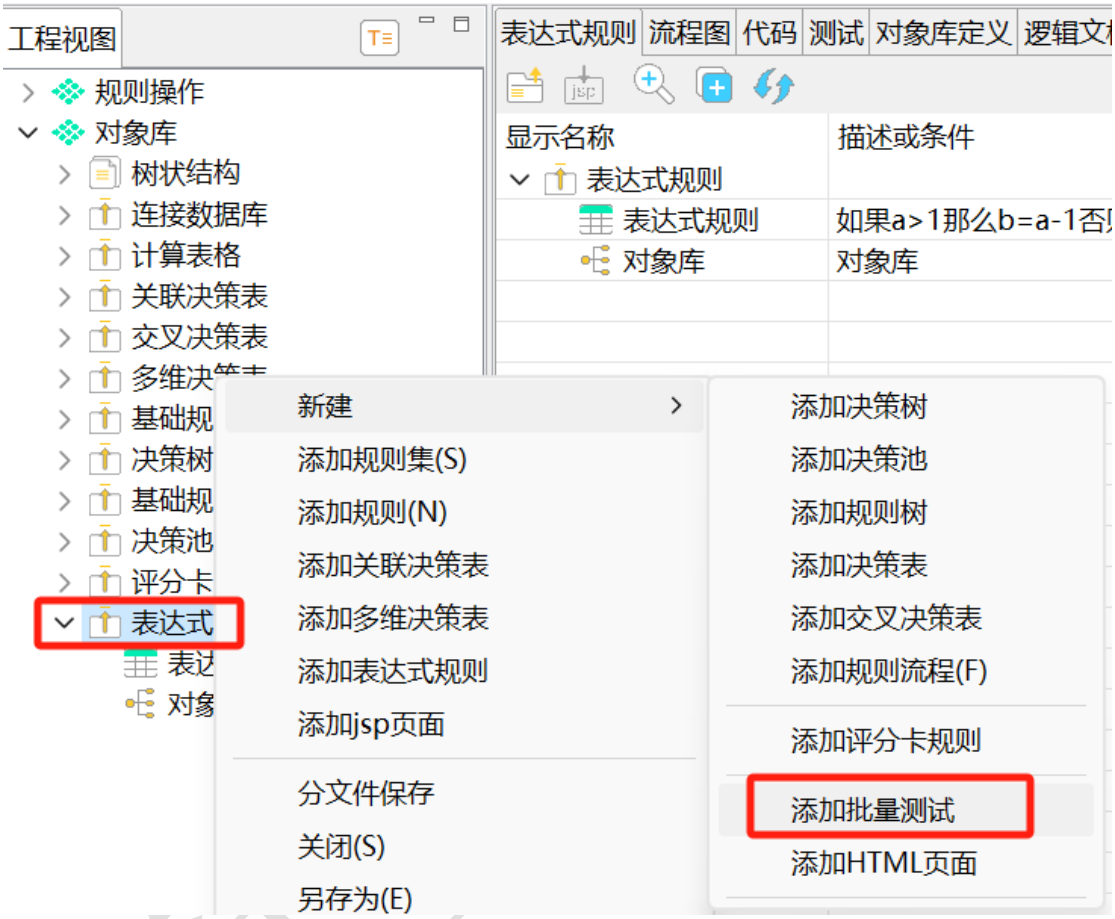
这样我们的表达式规则的逻辑就设置完成了，接下来我们要对规则包输出结果进行测

试。

4．测试

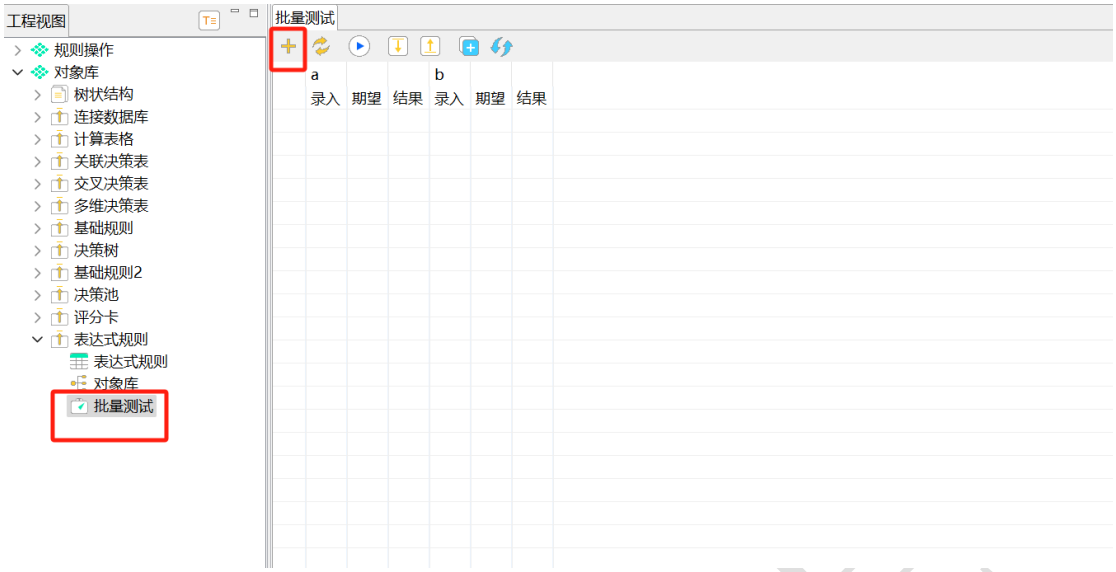
(1) 添加批量测试

我们在测试规则包的输出之前，我们要为规则包添加“批量测试”，然后我们可以在“批量测试”中添加测试用例进行测试。添加“批量测试”的步骤如下图所示：

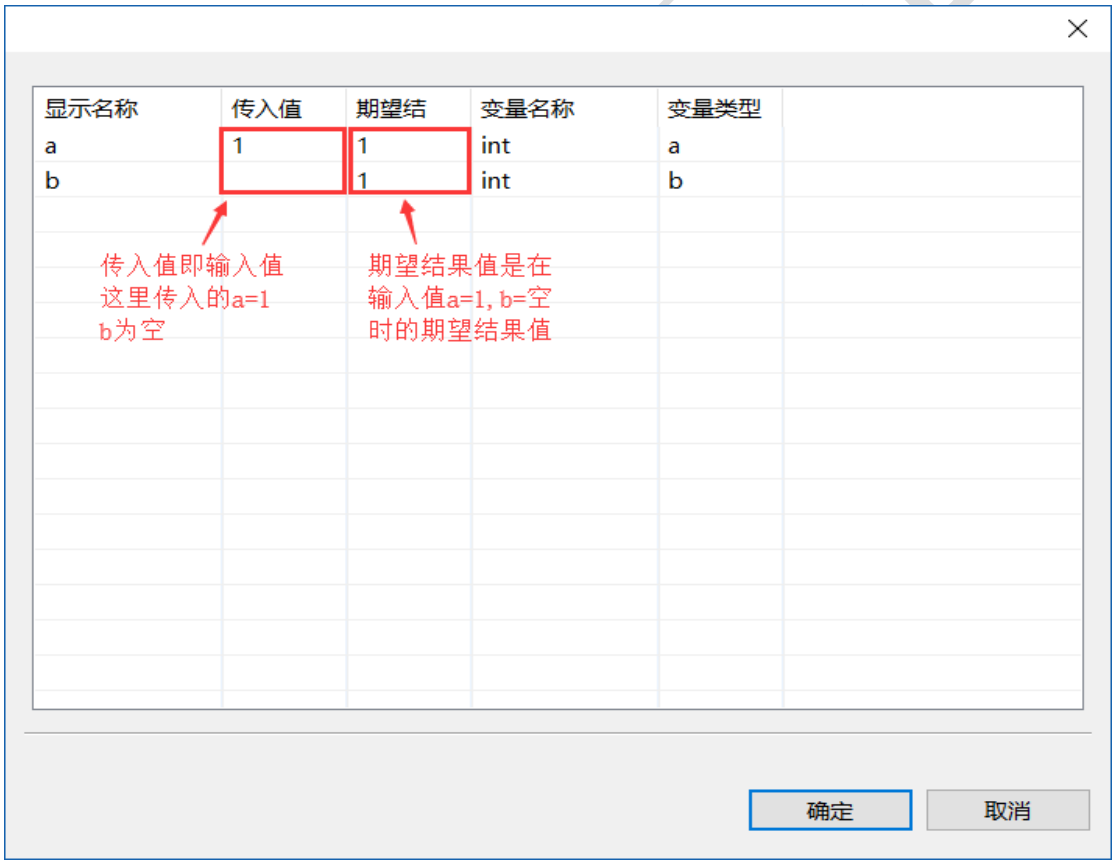


(2) 添加测试用例

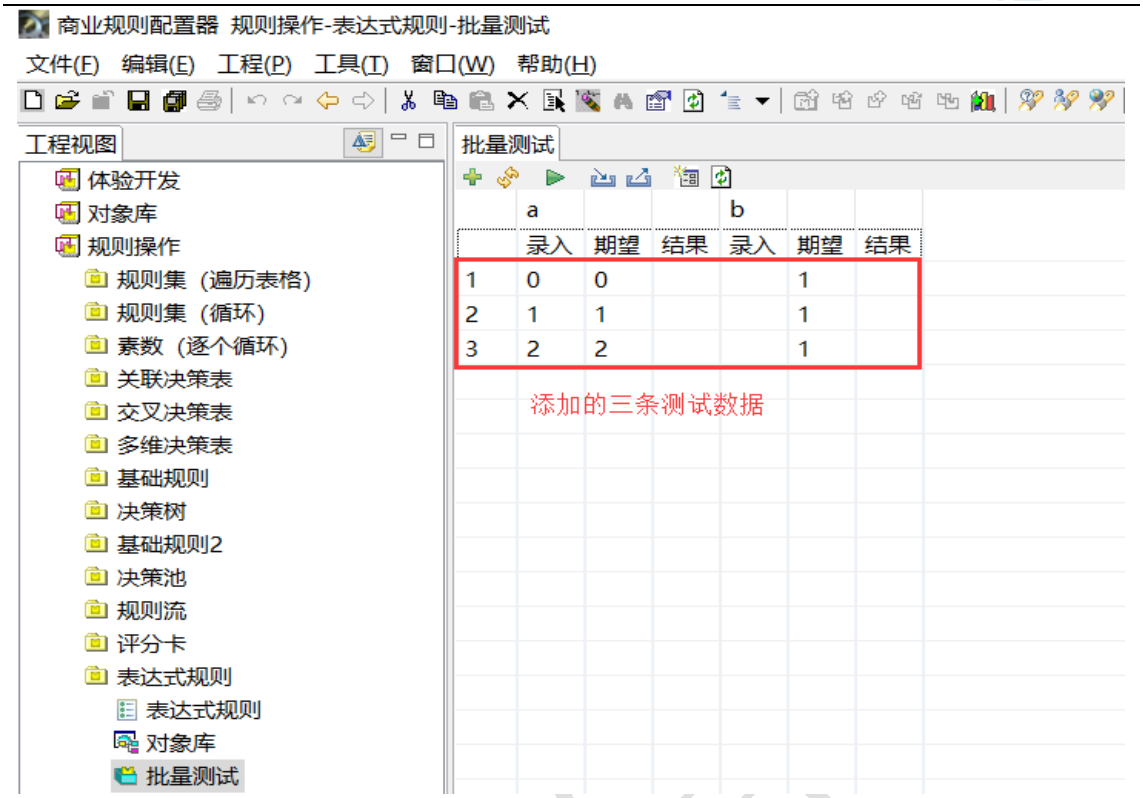
添加完了“批量测试”之后，我们需要给“批量测试”添加测试用例。添加过程如下所示：



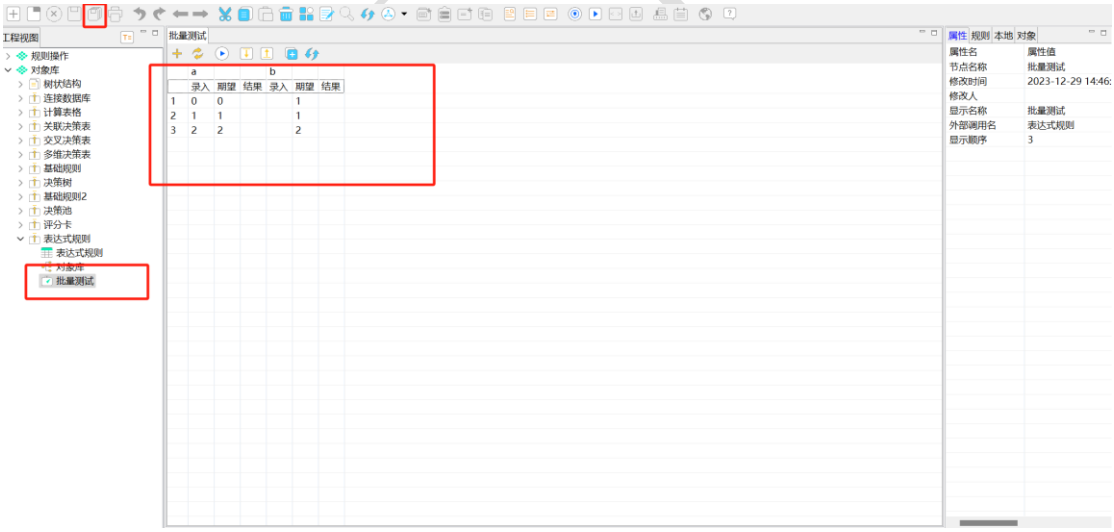
点击了“”按钮之后，我们需要在弹出页面中添加测试数据，添加过程如下：



确认之后，再次在批量测试中，按照上述步骤多添加几个测试用例。添加完成之后，批量测试的页面如下图：

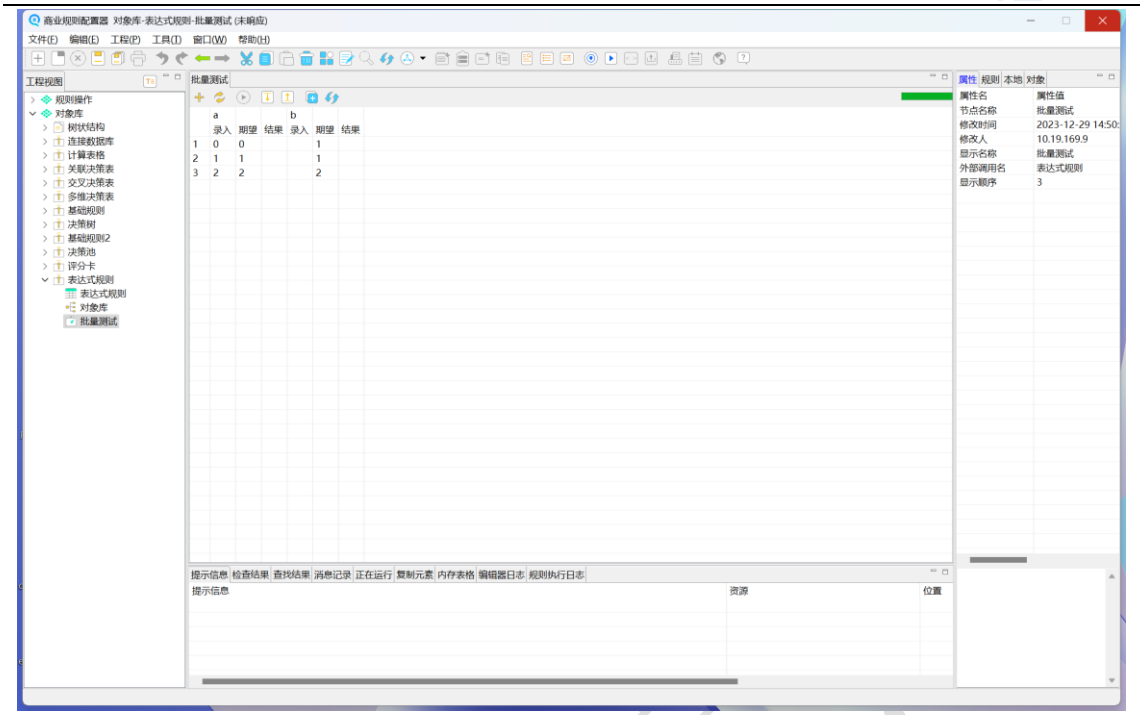


在测试用例添加完成之后，需要对规则包进行保存，选择“全部保存”按钮，这样规则包就自行保存并编译，其操作步骤如下图所示：



(3) 测试结果值

测试用例添加完成，规则包保存、编译之后，我们需要测试规则包是否满足这些测试用例的输出。测试的过程如下图所示：



3.9. 表达式表格

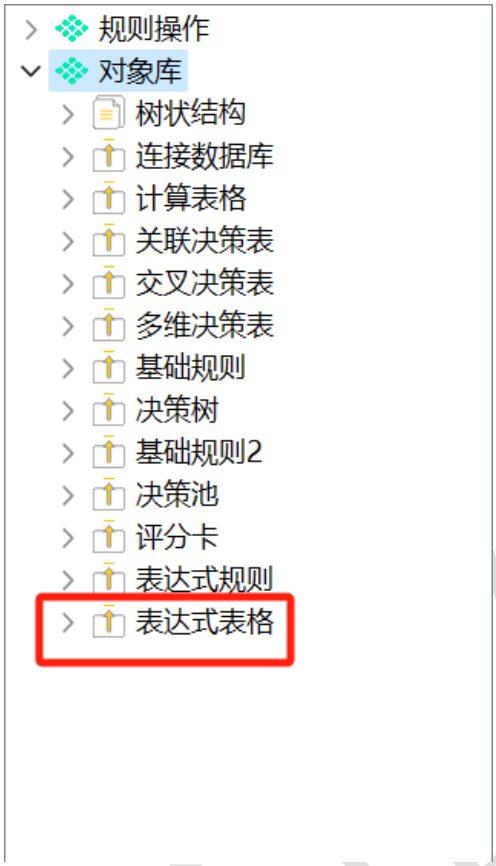
我们在对数据库的表进行操作的时候，通常采用的方式就是通过写 sql 对表进行增、删、改、查。若大量的查询都是在表中操作，就会大大增加表工作量，对表的性能会有影响。

规则引擎提供了内存表的概念，内存表就是一张虚拟表。我们可以把数据库中表的数据导入到内存表中，再对内存表进行操作。

表达式表格，实际上就是对内存表进行查询操作。在这个表达式中，我们可以设置不同的条件，查询对应的结果。

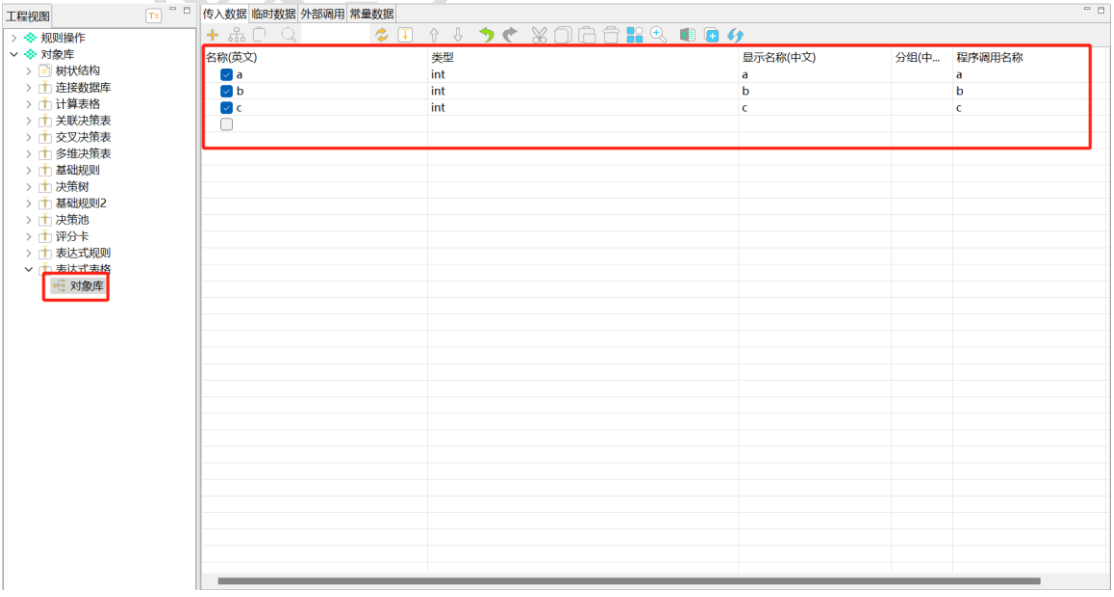
1. 创建规则包

在工程“规则操作”下新建规则包“表达式表格”：



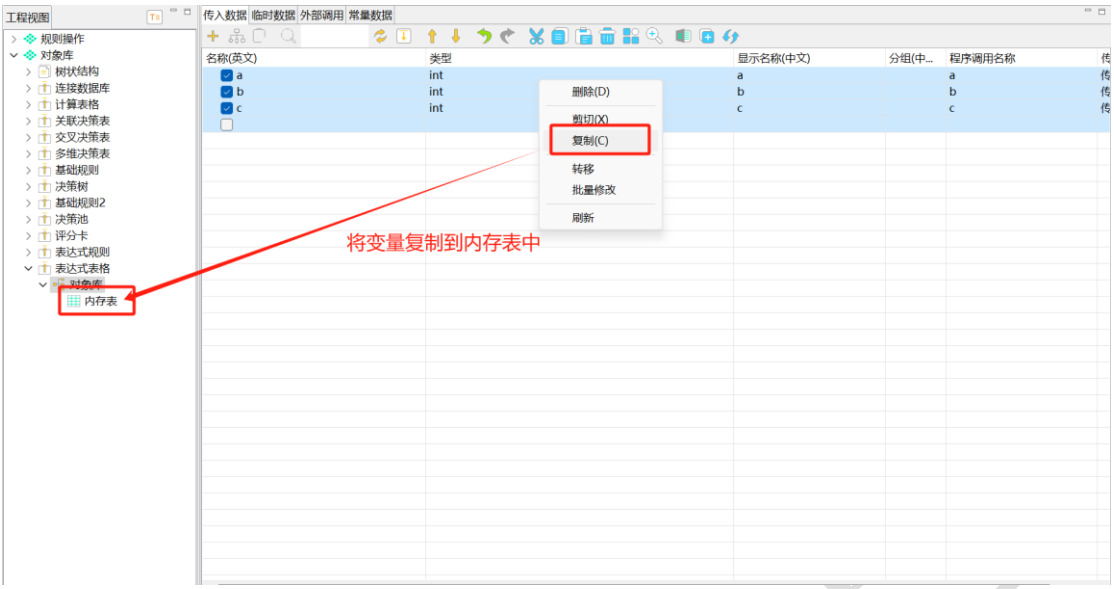
2. 定义变量

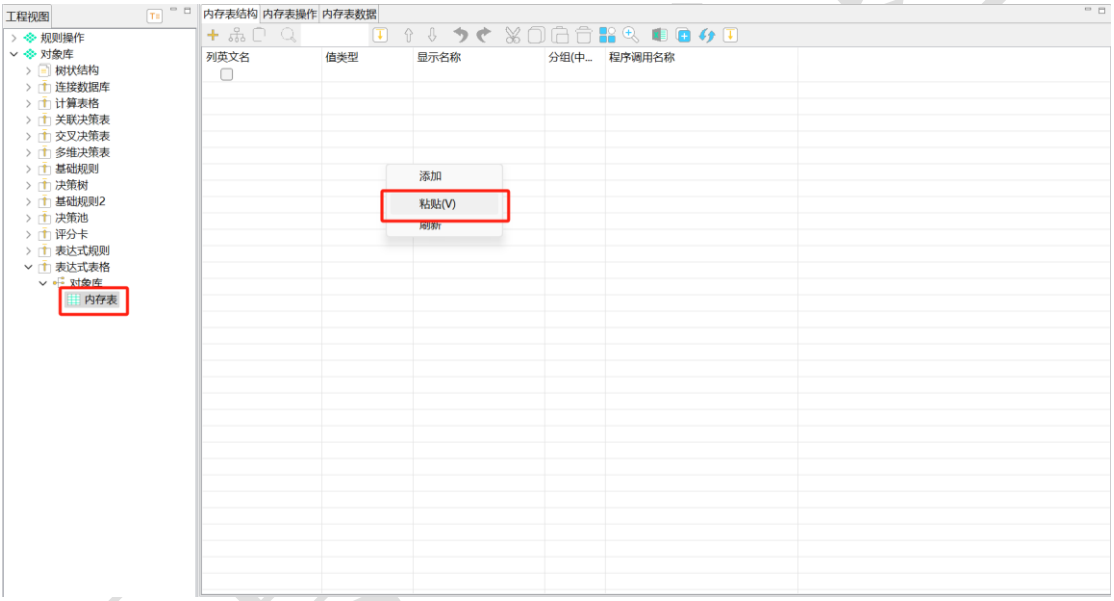
在该规则包中需要在对象库中，定义三个变量：a（a），b（b），c（c）。如下图所示：



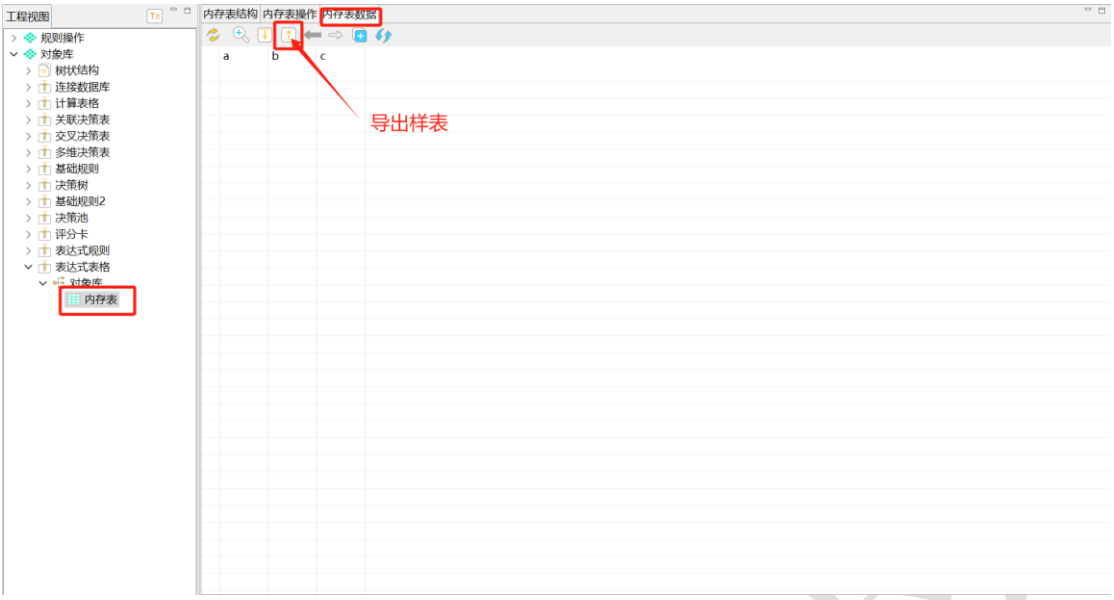
对象库中变量定义完成之后，需要添加一个计算表格“内存表”，并为该内存表定义变

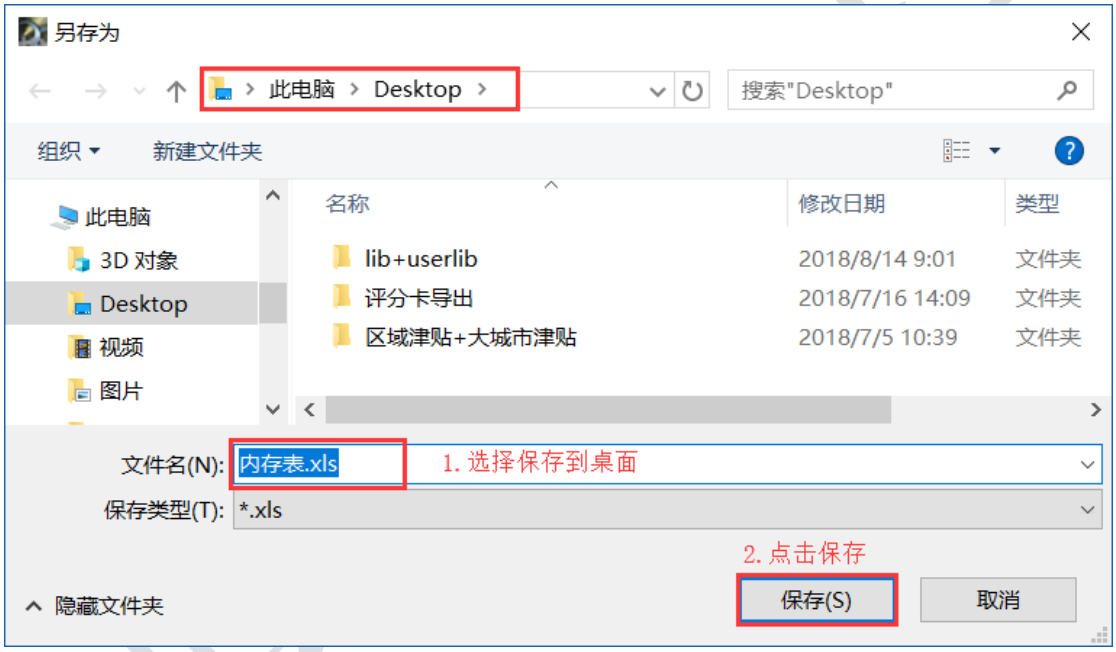
量。这里我们简单的做法就是把对象库中定义好的变量直接复制过来。步骤如下图所示：





粘贴完成之后，我们就把对象库的变量复制到内存表中了。我们接着步骤就是对内存表中的变量赋常量值，在规则配置器中，我们都是先把内存表导出到 excel 中，然后在 excel 中添加数据，然后再把 excel 导入到规则配置器的内存表中。需要注意的是这种方式给内存表赋值，内存表是作为一张常量表来使用；但是在实际运用过程中内存表往往只为暂存数据而使用。给内存表赋值流程如下图所示：

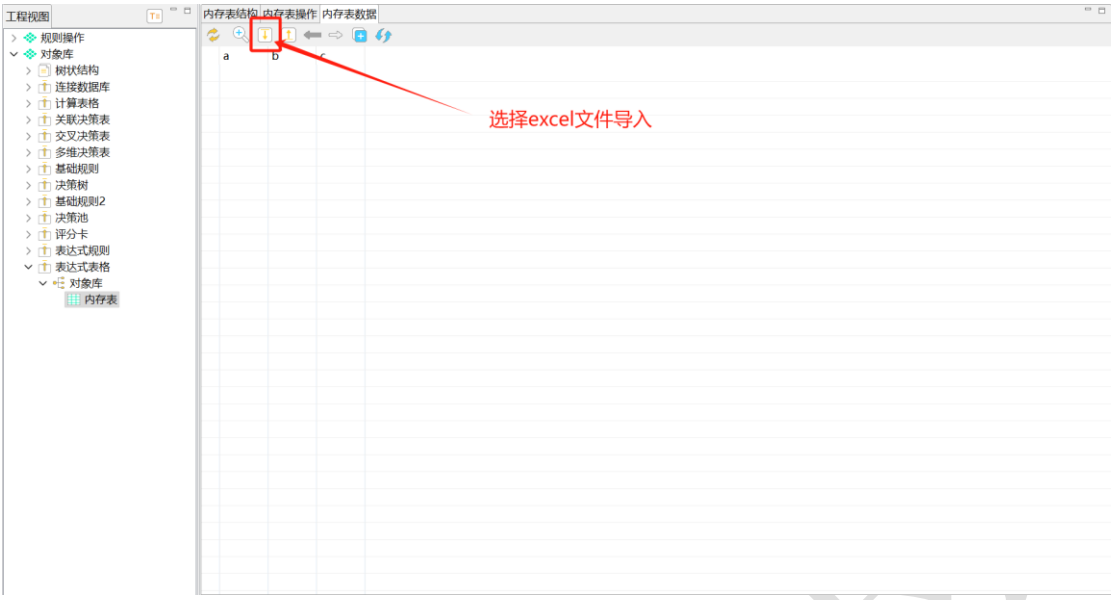




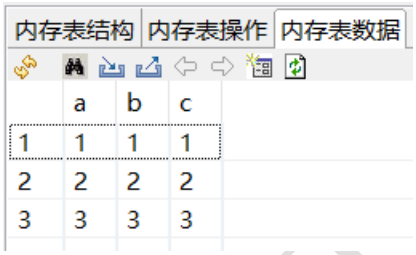
打开“内存表.xls”文件，添加如下数据：

	A	B	C	D
1	a	b	c	
2	1	1	1	
3	2	2	2	
4	3	3	3	
5				

数据添加完成之后，保存 Excel 文件，然后在规则配置器中将此 Excel 中的数据导入到内存表中，操作步骤如下图所示：

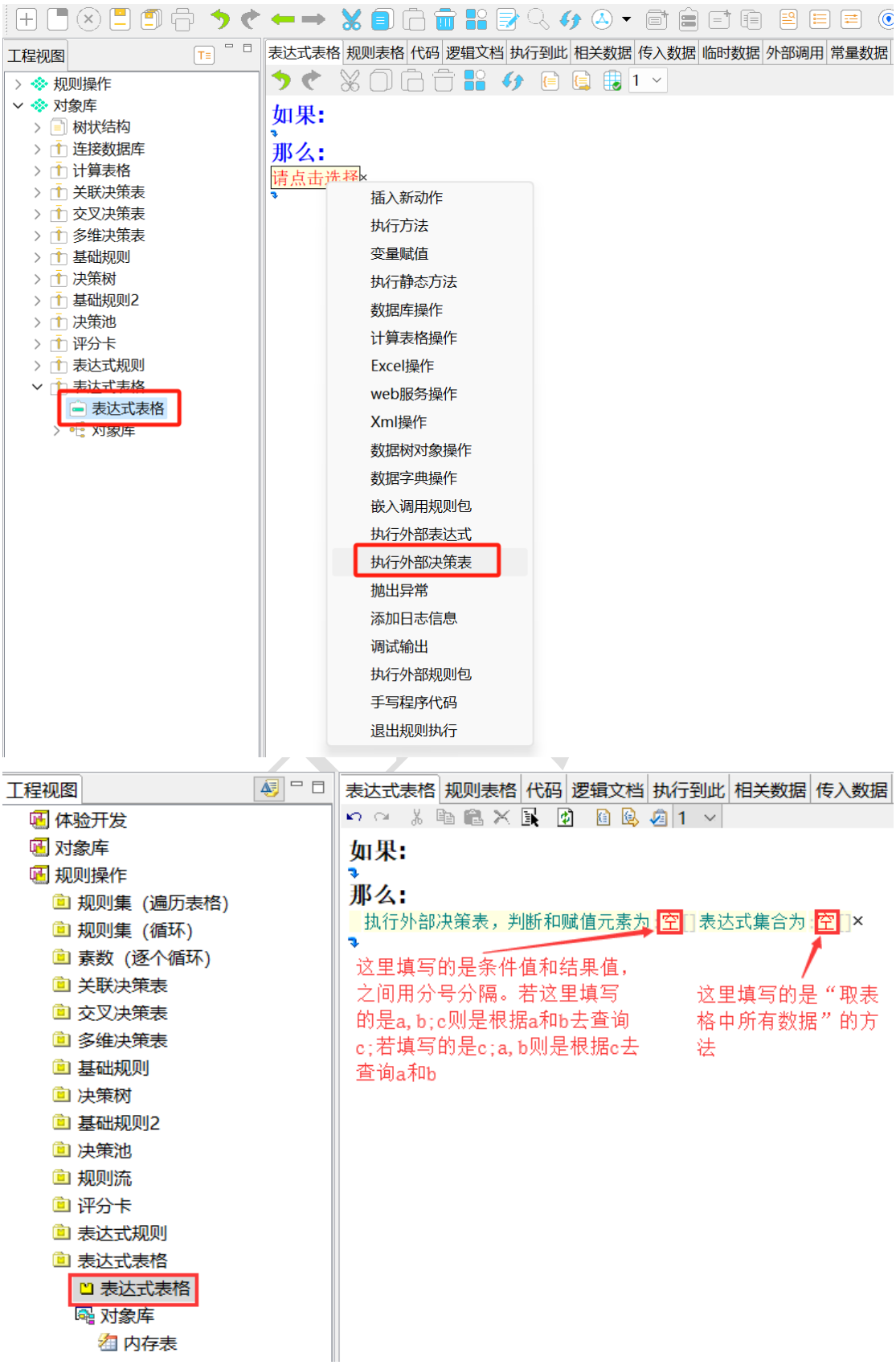


导入完成之后内存表中数据如下图所示：

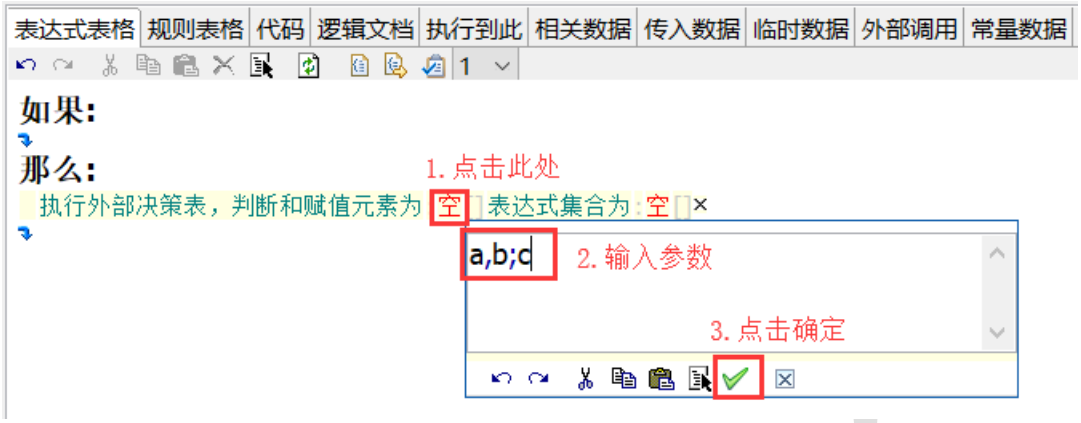


3. 编写规则

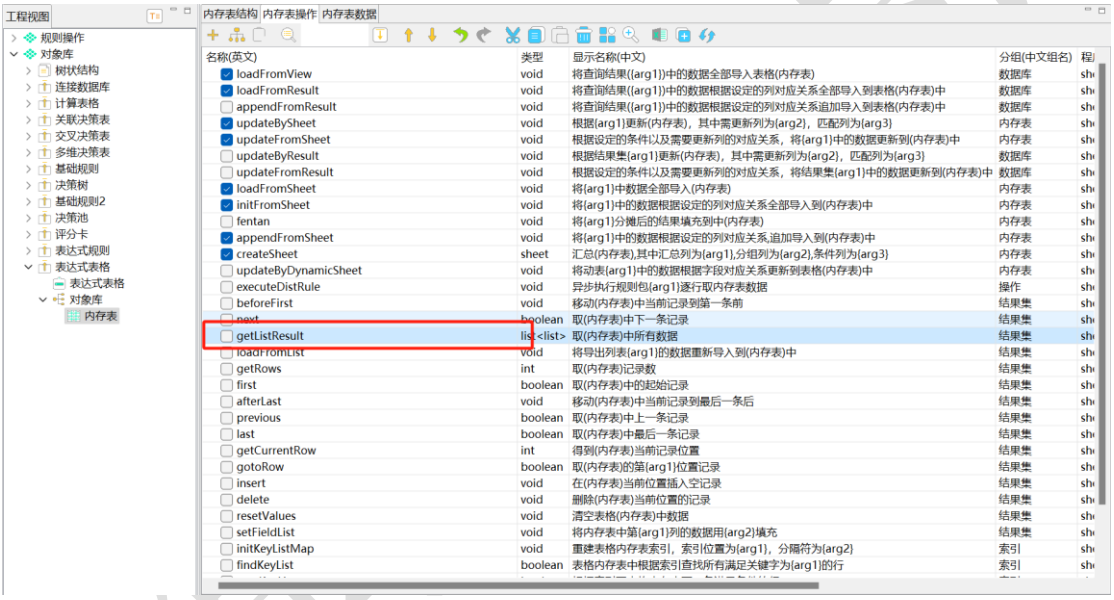
在规则包“表达式表格”中右键添加规则“表达式表格”，并在编辑窗口中添加如下规则逻辑：

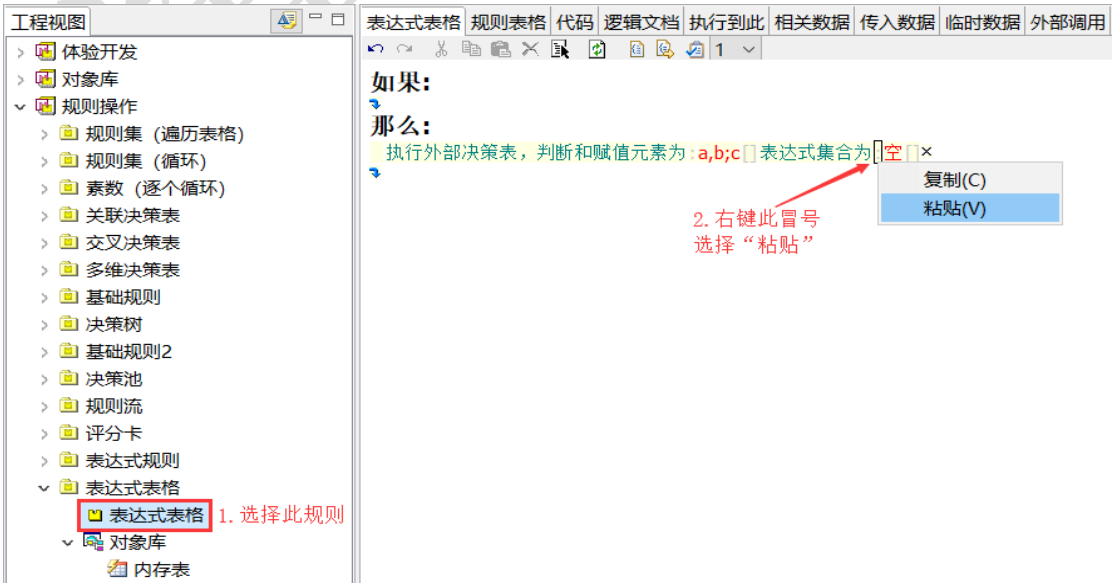


在这个表达式表格中，我们需要的是根据传入的变量“a”和“b“去查询对应“c”的值。因此，我们要在第一个“空”处，添加的内容如下图所示：



由于在第二个“空”是取去内存表中所有的数据，规则配置器在内存表中封装了对内存表操作的方法，所以这里要把该方法复制出来。其操作流程如下所示：



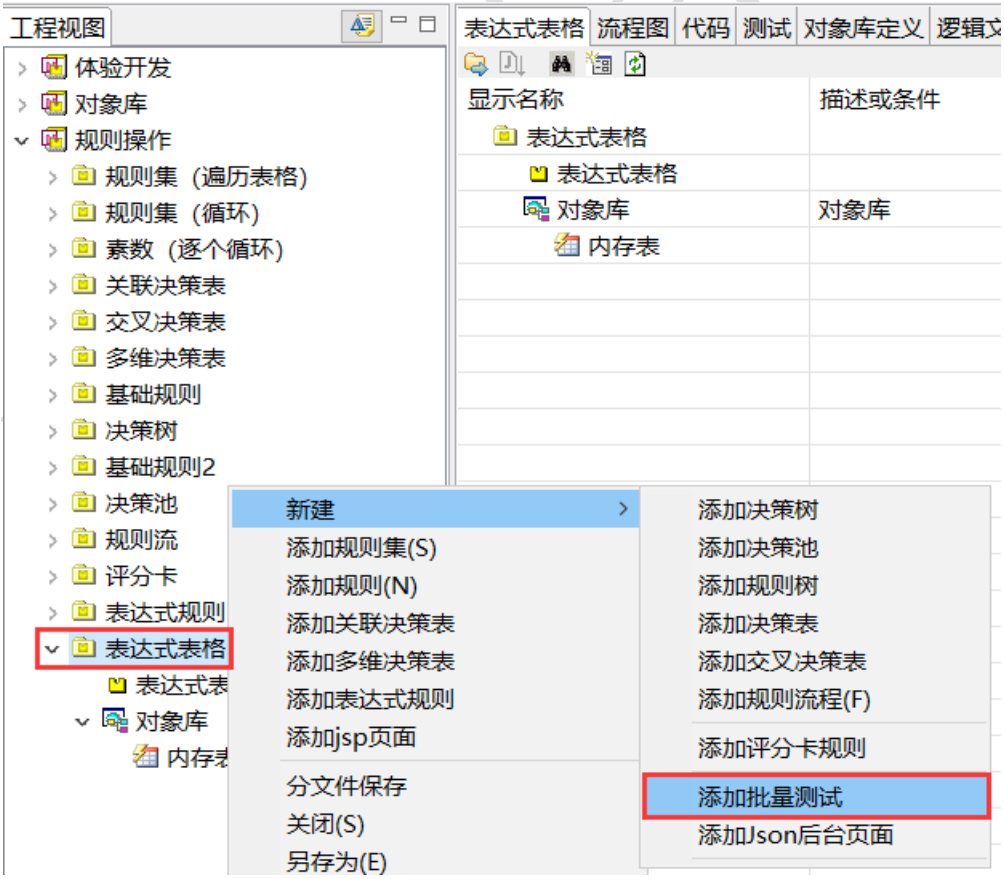


完成后规则如下图所示：

4. 测试

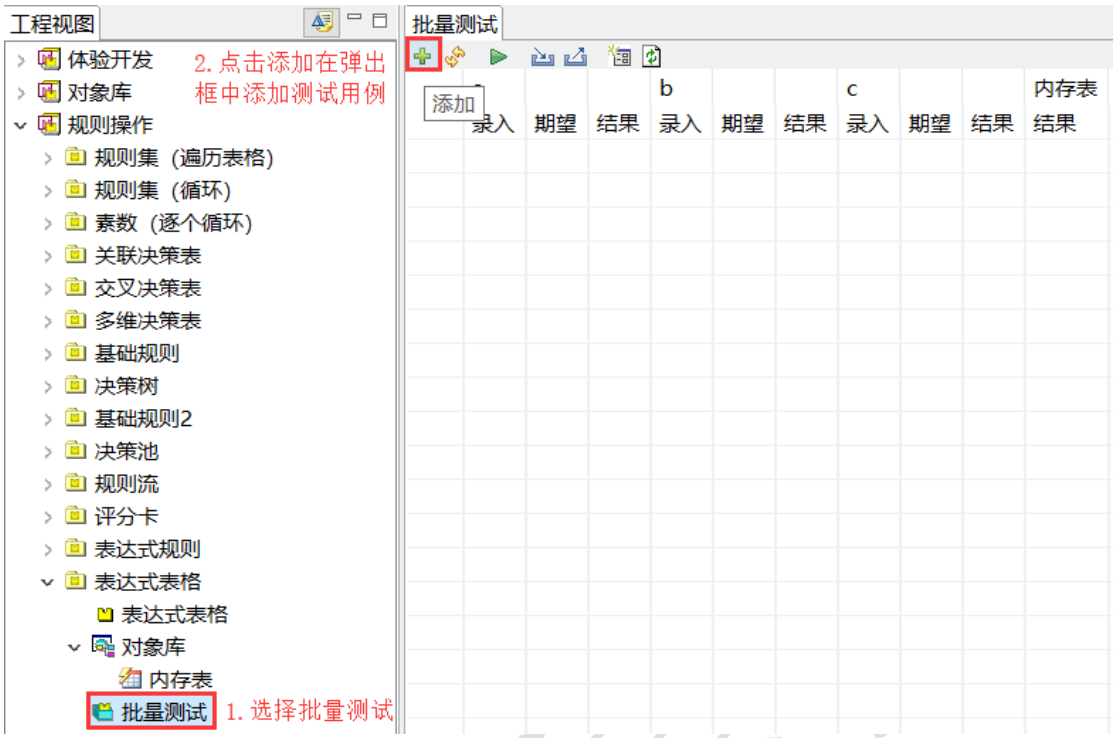
(1) 添加批量测试

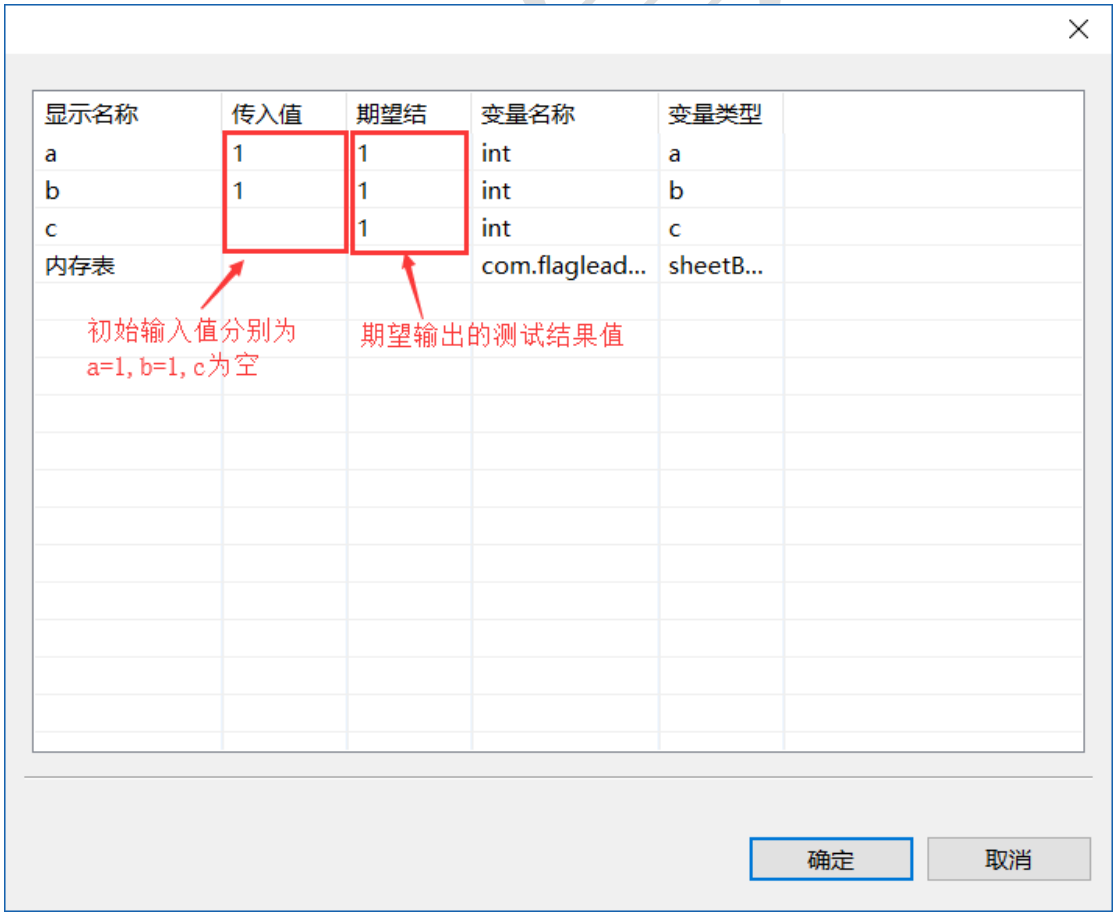
右键规则包“表达式表格”，选择新建→添加批量测试：



(2) 添加测试用例

添加完了“批量测试”之后，我们需要给“批量测试”添加测试用例。添加过程如下所示：





根据上述方法步骤，再次添加几条测试用例，添加完成如下图所示：

批量测试

	a			b			c			内存表
	录入	期望	结果	录入	期望	结果	录入	期望	结果	结果
1	1	1		1	1			1		
2	2	2		2	2			2		
3	3	3		3	3			3		

测试用例添加完成之后，点击“全部保存”按钮“”，保存并编译规则包。

(3) 执行批量测试

测试用例添加完成，规则包保存、编译之后，需要测试规则包是否满足这些测试用例的输出。若不满足：要么是规则编写的问题，要么是我们在输入期望值时存在错误，若满足，则测试完成。

注：我们在批量测试中，所有的测试结果都满足，并不代表该规则包一定正确。测试的过程如下图所示：

在此批量测试中，所有的期望值和结果值都相同，批量测试完成。