**易平台自定义组件实现**

目录

[目录 2](#_Toc424053440)

[1 组件类型 3](#_Toc424053441)

[2 单机Jar组件的实现 3](#_Toc424053442)

[3 Spark Jar组件的实现 7](#_Toc424053443)

[4 单机Shell的实现 10](#_Toc424053444)

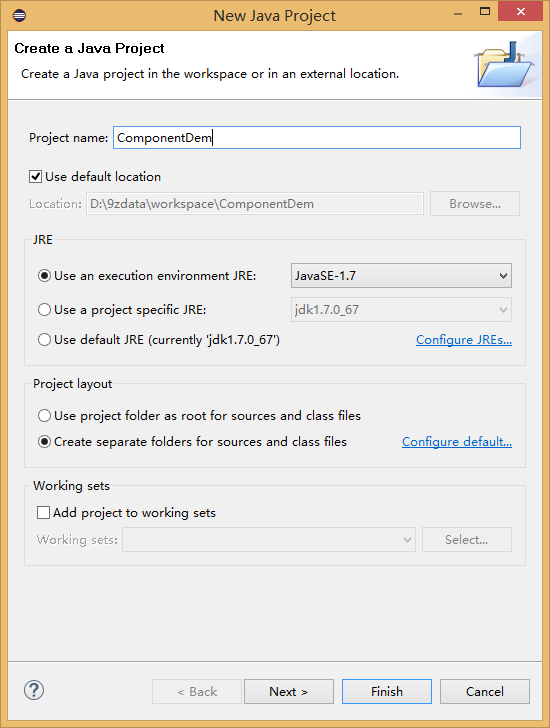
# 组件类型

易平台可以自定义三种类型的组件：

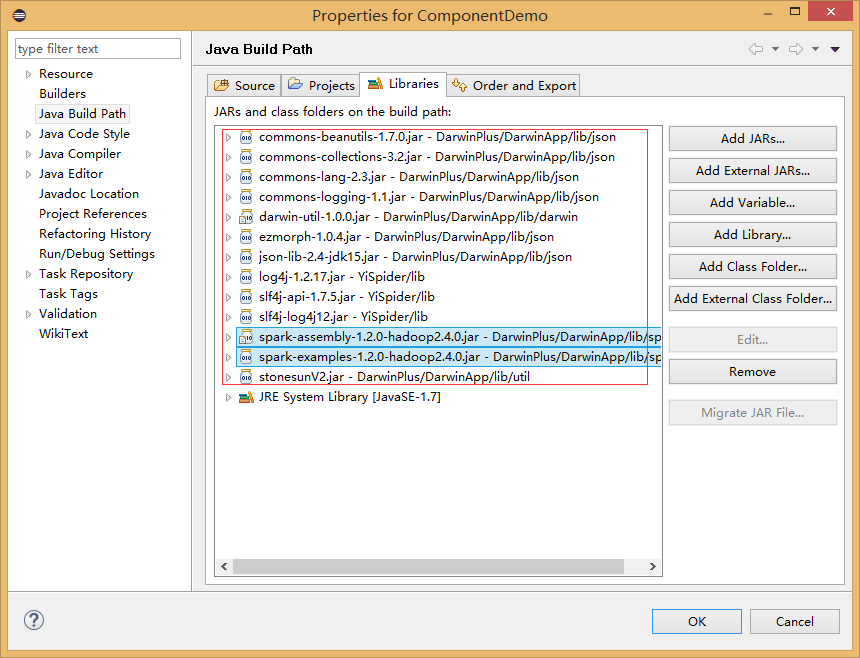
1. 单机Jar：单机运行的java程序。
2. Spark Jar：Spark程序。
3. 单机Shell：单机运行的shell脚本程序。

# 单机Jar组件的实现

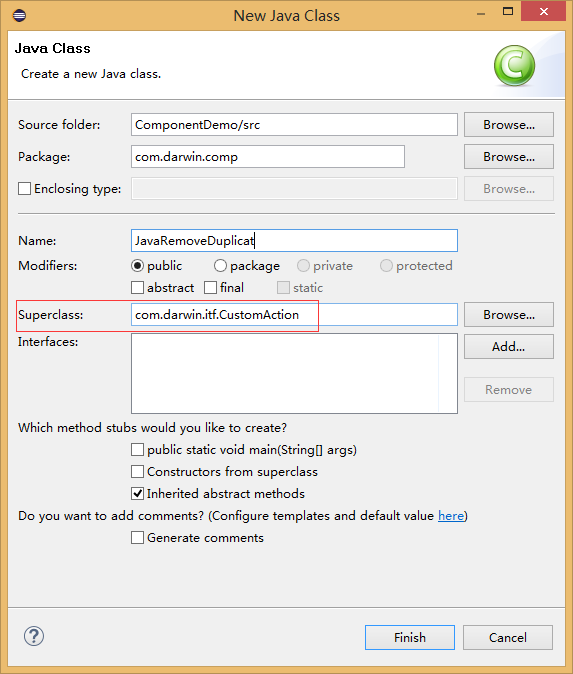
1. 创建java工程。示例工程名为ComponentDemo。



1. 下载自定义组件依赖的jar包（自定义组件依赖Jar包.zip），并通过build path加入到工程里面。



1. 创建一个继承com.darwin.itf.CustomAction的类（示例程序中的类名为：JavaRemoveDuplicate）。



1. 需要实现三个方法：

runAction:自定义组件的具体业务逻辑在该方法中实现，可以通过参数中的logger来打印日志。

getOutputFileName:如果组件需要将结果写入HDFS，最终的文件名需要通过该方法指定，例如：“rm\_dup\_result”。

terminateAction：该方法用来终止组件的，如果组件需要终止，而组件中用到了多线程或者循环，可以通过实现该方法来终止。

1. CustomAction提供的其他方法：

getCompId：获取组件ID。

getCompCustomParam：获取前台页面传入的参数，参数必须与“自定义组件新增编辑”页面的参数名称相同。

readLine：按行读取HDFS文件内容。

getOutPutPath：获取结果保存的HDFS路径。

writeLine：按行写入HDFS。

flushLineOutput：flush out put util.

getPeriodStr:获取当前组件所在组件所处的周期。

下面我们看一下组件的完整实现代码：

/\*\*

\* Java去重组件:单机Jar，需要继承CustomAction。</br>

\* 该示例程序实现了从HDFS获取数据，通过java程序去重，将结果保存到HDFS上。

\*/

**publicclass**JavaRemoveDuplicate**extends**CustomAction {

**privatestaticfinallong*serialVersionUID*** = 2461183955101761992L;

@Override

**publicint**runAction(CompLoggerInterfacelogger) **throws** Exception {

// 获取页面输入的参数，通过基类中的getCompCustomParam方法。

String inputPath = getCompCustomParam("inputPath"); // 输入路径。

Set<String>contentSet = **new**HashSet<String>();

// 使用基类中的readLine方法，按行读取HDFS数据。

logger.info("input path:" + inputPath);

**while** (readLine(inputPath, "UTF-8") != **null**) {

String line = readLine(inputPath, "UTF-8").trim();

logger.info(line);

contentSet.add(line);

}

// 使用基类中的writeLine方法，按行写入HDFS。

**for** (String line : contentSet) {

writeLine(line + "\n");

}

// 使用logger可以在页面中打印日志。

logger.info("自定义组件实现完成！");

**return** 0;

}

/\*\*

\* 如果运算结果需要落地到HDFS，这里需要给出落地的文件名</br>

\* 因为页面只给出到最终结果说在的路径。</br>

\*/

@Override

**public** String getOutputFileName() {

// 可以指定一个固定名称，也可以按照周期时间规则生成文件名。

**return**"rm\_dup\_result";

}

@Override

**publicboolean**terminateAction() **throws** Exception {

// 协助终止for循环、终止线程（池）

**returnfalse**;

}

}

1. 打包上传：将自定义组件打成jar包（不要包含所依赖的jar包），如果自定义组件还依赖了除“自定义组件依赖Jar包.zip”中之外的其它jar包，则将那些jar包同组件打成的jar包一起打成一个zip包上传；如果没有依赖其他jar包，则只上传组件打成的jar包即可。

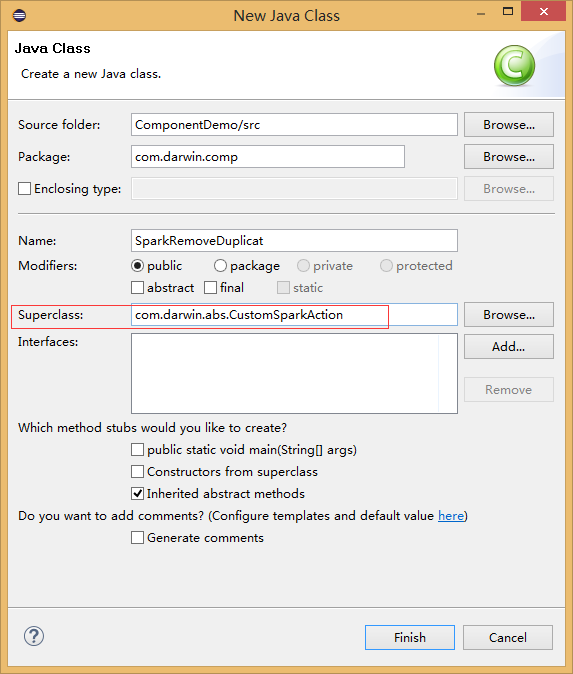


该组件只需要依赖“自定义组件依赖Jar包.zip”中的jar包即可，所以只需上传该组件打成的jar包即可。另外添加的参数里面的参数名称inputPath一定要与代码中getCompCustomParam方法的参数保持一致。注意类名不要写错，一定要是继承CustomAction的那个类。

1. 自定义组件创建完毕，可以创建一个流程来测试一下了。

# Spark Jar组件的实现

1. 为该组件创建一个java工程。这里还是使用ComponentDemo工程。
2. 下载自定义组件依赖的jar包（自定义组件依赖Jar包.zip），并通过build path加入到工程里面。（同单机Jar步骤2）。
3. 创建一个继承com.darwin.abs.CustomSparkAction的类（示例程序中的类名为：SparkRemoveDuplicate）。



1. 需要实现一个方法：

runAction:自定义组件的具体业务逻辑在该方法中实现，可以通过参数中的logger来打印日志。

1. CustomSparkAction提供的其他方法：

getSC：获取SparkContext。

getCompId：获取组件ID。

getJobId：获取Spark任务ID。

getCompCustomParam：获取前台页面传入的参数，参数必须与“自定义组件新增编辑”页面的参数名称相同。

getHadoopRdd：根据HDFS路径获取RDD。

storeRDD：将结果保存到HDFS。

transRDD：转换RDD，将每行数据按照分割符或者正则表达式分割。

组件的完整代码：

/\*\*

\* Spark去重组件：Spark Jar，需要继承CustomSparkAction。

\* 该示例程序使用Spark去重。

\*/

**public class** SparkRemoveDuplicate **extends** CustomSparkAction {

**privatestaticfinallong**serialVersionUID = 9123555311896322717L;

@Override

**public int** runAction(CompLoggerInterface logger) **throws** Exception {

// 获取页面输入的参数，通过基类中的getCompCustomParam方法。

String inputPath = **this**.getCompCustomParam("inputPath"); // HDFS路径

// Spark组件去重逻辑。

JavaRDD<String>resultRdd = getHadoopRdd(inputPath, logger)

.mapToPair(**new**PairFunction<String, String, Integer>() {

**privatestaticfinallong**serialVersionUID = 1L;

@Override

**public** Tuple2<String, Integer> call(String t) **throws** Exception {

**returnnew** Tuple2<String, Integer>(t, 1);

}

}).reduceByKey(**new** Function2<Integer, Integer, Integer>() {

**privatestaticfinallong**serialVersionUID = 7727986215116381837L;

@Override

**public** Integer call(Integer v1, Integer v2) **throws** Exception {

**return** 1;

}

}).map(**new** Function<Tuple2<String,Integer>, String>() {

**privatestaticfinallong**serialVersionUID = -4891649426560670938L;

@Override

**public** String call(Tuple2<String, Integer> v1) **throws** Exception {

**return** v1.\_1;

}

});

// 调用基类的storeRDD方法，将结果保存到HDFS上。

storeRDD(resultRdd);

**return** 0;

}

}

1. 打包上传。过程同单机Jar包步骤６一致，只需要将类型改为SparkJar。
2. 创建流程进行测试。

# 单机Shell的实现

编写shell脚本程序，主程序会传入两个参数，第一个参数为用户配置的参数文件，第二个参数为周期串（以test.sh为例）

shell组件示例代码:

# !/bin/bash

source /home/yimr/.bash\_profile

arg\_file=$1

period\_str=$2

#从配置正则里面，提取配置的输出目录

target\_dir=$(cat $arg\_file | sed 's/.\*hdfs\",\"value\":\"\([^\"]\*\)\".\*/\1/g')/$period\_str

hadoop fs –test –d $target\_dir

if [ $? –ne 0 ];then

hadoop fs –mkdir -p $target\_dir

fi

hadoop fs –rm $target\_dir/$(basename $arg\_file)

hadoop fs –put $arg\_file $target\_dir/

exit 0

将shell脚本和其依赖的文件一起打包成zip文件（如test\_comp.zip）

打包上传。过程同单机Jar包步骤６一致，不需要填类名，而要填写入口文件test.sh（即shell执行程序），同时将类型改为shell。

创建流程进行测试。