Macro

Macro is a kind of function which can be reused over the time and again.
It is introduced in Pig 0.9 version to make the pig code more modular and sharable.
We will use employee.txt to understand the macro.

Let's try to find out all TCS employees without using Macro.

```
grunt>
A= LOAD '/home/hduser/employee.txt' using PigStorage(',')
    AS (org:chararray,name:chararray,salary:float);
B= FILTER A by org=='TCS';
dump
```

You can move the code to filter the data into MACROS as shown below

```
DEFINE getTCSEmployee(relation,column) RETURNS x{
$x = FILTER $relation BY $column=='TCS';
};
```

Now your script becomes

```
A= LOAD '/home/hduser/employee.txt' USING PigStorage(',') AS
  (org:chararray,name:chararray,salary:float);
B= getTCSEmployee(A,org);
dump
```

Note: You cannot pass 'TCS' as parameter to macro to make it dynamic. For that you need to move to UDFs. You can only pass the relation name, column name as macro parameter.

The below code is invalid.

```
DEFINE getEmpByCompany(relation,column,companyName) returns x{
$x = filter $relation by $column==$value;
};
```

Externalize your Macro

Instead of defining the macro inline, you can better create a separate macro file and define your macro there as shown below..

```
$ vim /home/hduser/myMacro.macro
--write the below piece of code
DEFINE getTCSEmployee(relation,column) RETURNS x{
$x = FILTER $relation BY $column=='TCS';
};
```

Import the above macro in your Pig Terminal

```
grunt> IMPORT '/home/hduser/myMacro.macro';
```

Note: Before executing the above command, please quit the pig terminal and login again to pig, else you will get the below error. Because you already have getTCSEmployee macro defined in the session and your macro file also has the same macro name

```
2017-01-28 01:45:49,685 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1200: <at
/home/hduser/myMacro.macro, line 1> null. Reason: Duplicated macro name
'getTCSEmployee'
Details at logfile: /home/hduser/pig_1485595589168.log
```

Use the imported macro as we have done previously.

```
A= LOAD '/home/hduser/employee.txt' USING PigStorage(',') AS
  (org:chararray,name:chararray,salary:float);
B= getTCSEmployee(A,org);
dump
```

You can instead write the whole command in the pig script file and execute them

```
$ vim /home/hduser/getTCSEmp.pig
--write the below piece of code
IMPORT '/home/hduser/myMacro.macro';
A= LOAD '/home/hduser/employee.txt' USING PigStorage(',') AS
(org:chararray,name:chararray,salary:float);
B= getTCSEmployee(A,org);
dump
```

Execute the script now

```
$ pig -x local -f /home/hduser/getTCSEmp.pig
```

Chapter 21. JUnit Your Pig Script using PigUnit

In real time, when we have a pig script doing complex job with lot of combinations, it's really important for us to test it with varieties of Input data so that we are sure that our pig script will not break in any case.

Steps:

1. **Create word1.txt inside /home/hduser**

```
$vim /home/hduser/word1.txt
apple apple ball
apple apple ball
apple ball ball
apple ball
apple
ball
```

2. **Create Pig Script that takes word1.txt as input and calculates the word count.**

```
$ vim /home/hduser/wordcount.pig
inp = LOAD  '/home/hduser/word1.txt'  AS (line:chararray);
words = FOREACH inp GENERATE FLATTEN(TOKENIZE(line))  AS word;
grpd  = GROUP words BY word;
cntd  = FOREACH grpd GENERATE group, COUNT(words);
ord = ORDER cntd by $0  ASC;   --we want  (apple,7) then (ball,6) and not the other way
dump ord;
```

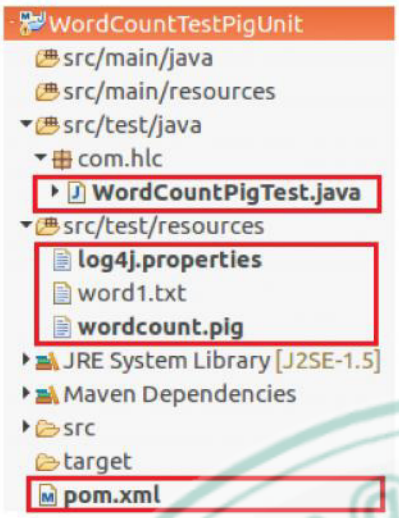
3. **Execute the above script and check the output**

```
$ pig -x local /home/hduser/wordcount.pig
--outputs the below
(apple,7)
(ball,6)
```

4. **Create a Maven Project with below co-ordinates**

   groupId org.hlc

   artifactId WordCountTestPigUnit



5. **Add the below dependency in the pom.xml. You need to add the below code directly under <project> tag.**

```
<properties>
    <hadoop-version>2.7.2</hadoop-version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>${hadoop-version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
        <version>${hadoop-version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-common</artifactId>
        <version>${hadoop-version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.pig</groupId>
        <artifactId>pig</artifactId>
        <classifier>h2</classifier>
        <version>0.16.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.pig</groupId>
        <artifactId>pigunit</artifactId>
        <version>0.16.0</version>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
```

```
        </dependency>
    </dependencies>
```

Note:

`<classifier>h2</classifier>` for pig means fetch pig-0.16 designed for hadoop2.Not including this will cause the test case to fail with the below error.

```
org.apache.pig.impl.logicalLayer.FrontendException: ERROR 1066: Unable to open iterator for alias ord.
Caused by: java.io.IOException: Couldn't retrieve job.
```

`<scope>test</scope>` Means this jar is just needed while running the test case, and really not required when building the project.

6. **Copy word1.txt ,wordcount.pig prepared in step 1 and step 2 and paste it to WordCountTestPigUnit/src/test/resources**

7. **Edit Your wordcount.pig script to point to word1.txt present in test/resources.**
   Please check the very first line of wordcount.pig present in step-2.

```
inp = LOAD '/home/hduser/word1.txt' AS (line:chararray);
```

It is pointing to /home/hduser/word1.txt. But now we have already copied the word1.txt into test/resources folder. Let's point to the word1.txt present in test/resources folder and not to the one present in /home/hduser

```
inp = LOAD 'src/test/resources/word1.txt' AS (line:chararray);
```

8. **Create log4j.properties in WordCountTestPigUnit/src/test/resources with the below configuration.**

```
# Set root logger level to DEBUG and its only appender to A1.
log4j.rootLogger=DEBUG, myConsoleAppender
# myConsoleAppender is set to be a ConsoleAppender.
log4j.appender.myConsoleAppender=org.apache.log4j.ConsoleAppender
# myConsoleAppender uses PatternLayout.
log4j.appender.myConsoleAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.myConsoleAppender.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n
```

9. **Create Your Java Test Class under the src/test/java**

```
package com.hlc;

import java.io.IOException;
import org.apache.pig.pigunit.PigTest;
import org.apache.pig.tools.parameters.ParseException;
import org.junit.Test;

public class WordCountPigTest {
    private static String output[] = { "(apple,7)", "(ball,6)" };

    @Test
```

```
  public void testWordCount() throws IOException, ParseException {
   PigTest pigTest = new PigTest("src/test/resources/wordcount.pig");
   pigTest.assertOutput("ord", output);
    }
}
```

Note: The output [] is the bag that you are expecting to get when your wordcount.pig script gets executed. You will get error if you interchange the order of the output.ie if you specify (ball,6) and then (apple,7)

In last line, ord is the relation that you are dumping from your script file. Please check step 2.

10. Run Your Test Class.

Right click on the Java class->Run As-> JUnit Test. It will show you JUnit view with Green colour indicating your Test is successful.