CentOS6.7: CDH5.5: Spark Jobs

[02/20/2016, 20:47 PM, Popova, Marina]

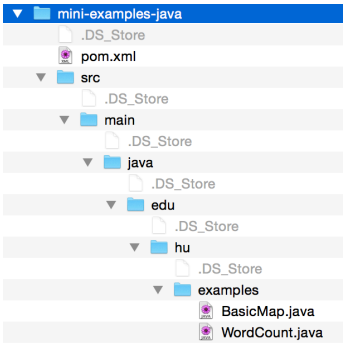
Java Spark applications development

Table Of Content:

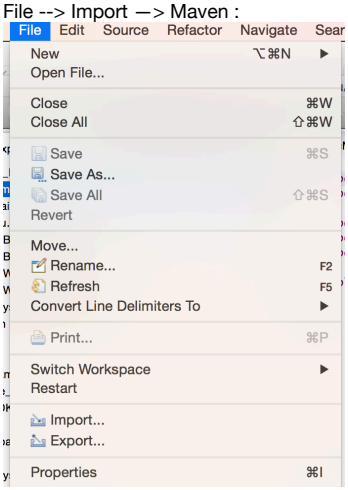
1. Developing and running Java Spark apps in Eclipse
2. Creating JAR with Maven on Mac
3. Compiling and building Spark Java jobs on CentOS6.7 VM

Developing and Running Java Spark apps in Eclipse

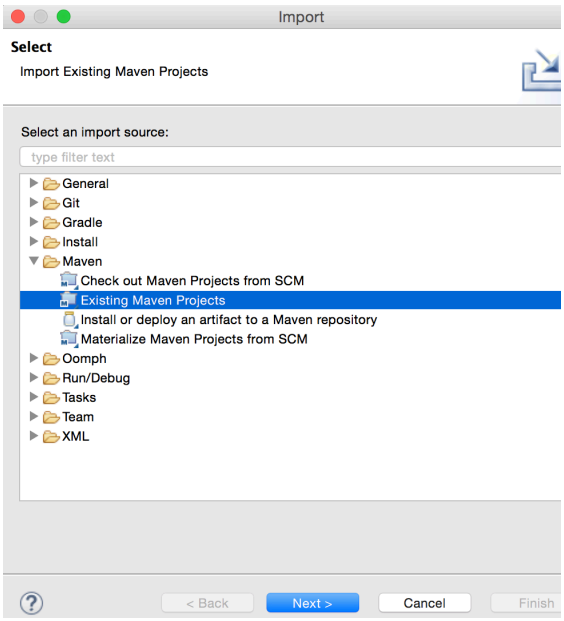
Create a directory structure of your Maven Spark project - using the provided
mini-examples-java.tar as the starting point: (ignore .DS_Store files that show up below - it is MAC thing….)



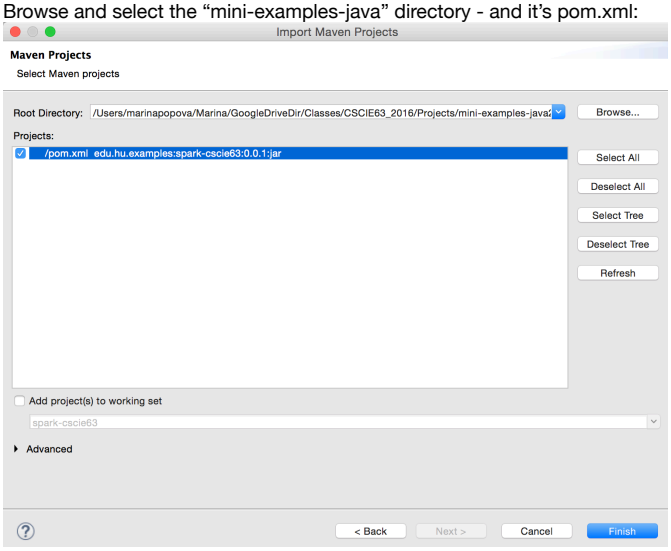
In Eclipse, create a new project as following:
File --> Import —> Maven :



select “Import Existing Maven Projects"

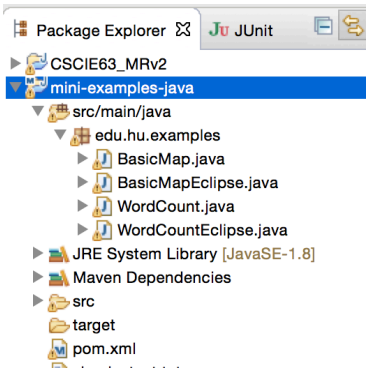


Browse and select the “mini-examples-java” directory - and it’s pom.xml:



Click “Finish"

You will have a new project created in Eclipse that will look similar to:

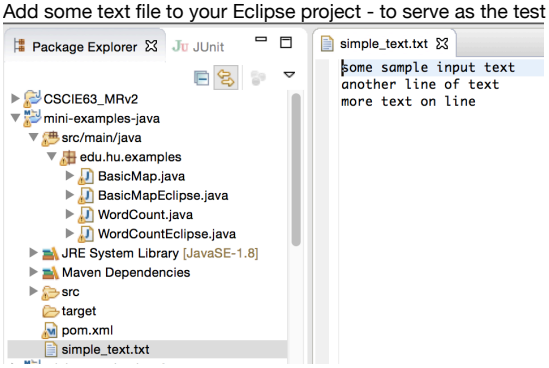


To run the WordCount example from Eclipse - the JavaSparkContext creation has to be modified a bit to provide additional parameters :

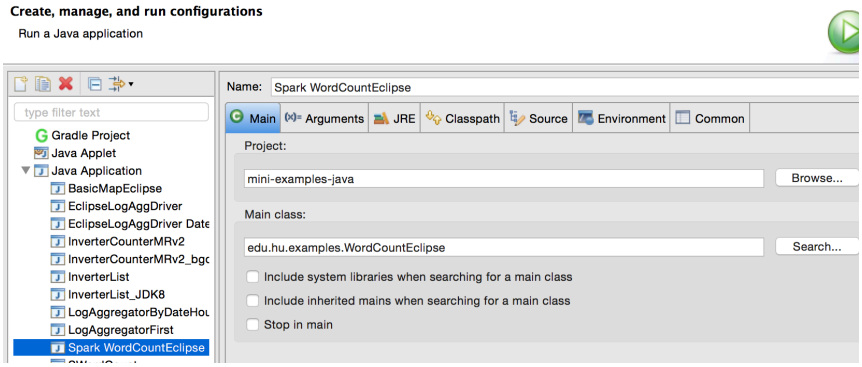
```
JavaSparkContext sc = new JavaSparkContext("local", "WordCountJavaApp");
```

The new class, WordCountEclipse.java has the change.

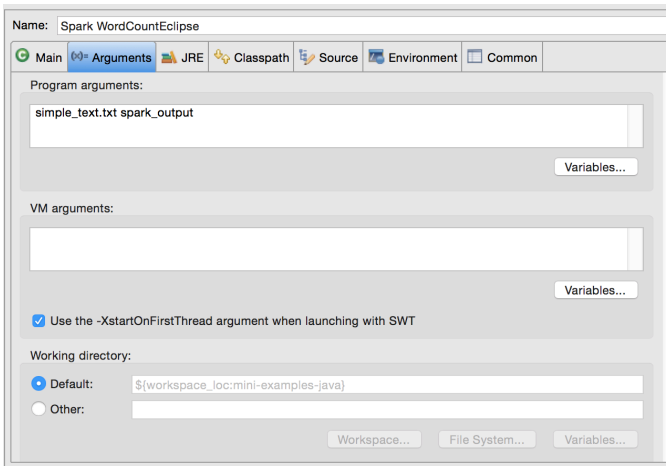
Add some text file to your Eclipse project - to serve as the test input for the word count class:



To run it - create a new Run Configuration in Eclipse:

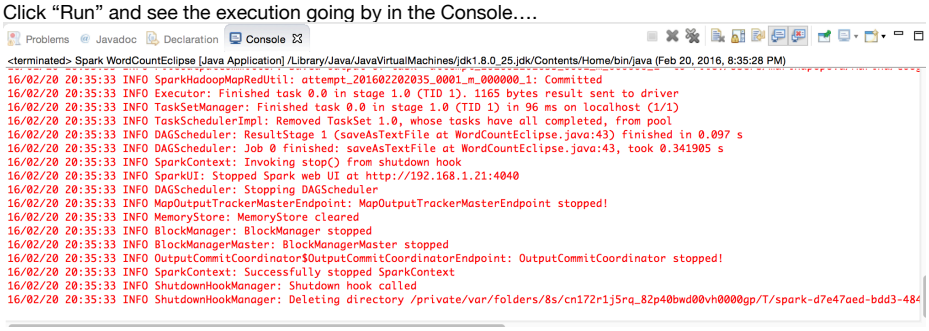


and set program arguments:

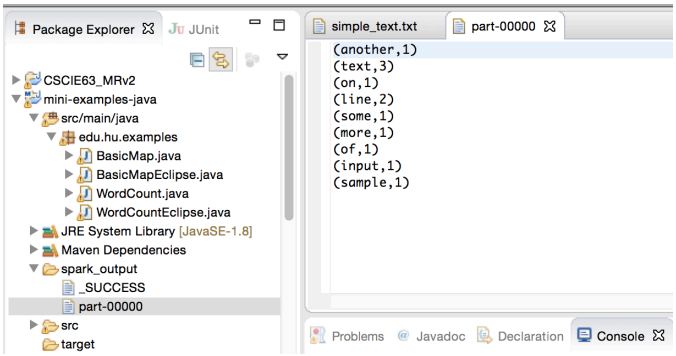


- simple_text.txt - is your test input
- spark_output - name of the directory for Spark output (will be created in the Eclipse project - make sure it does not exist before each run, just like the output dir for MR jobs)

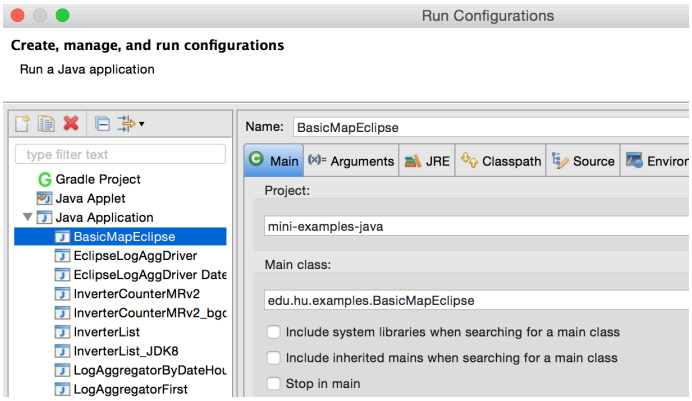
Click “Run” and see the execution going by in the Console….



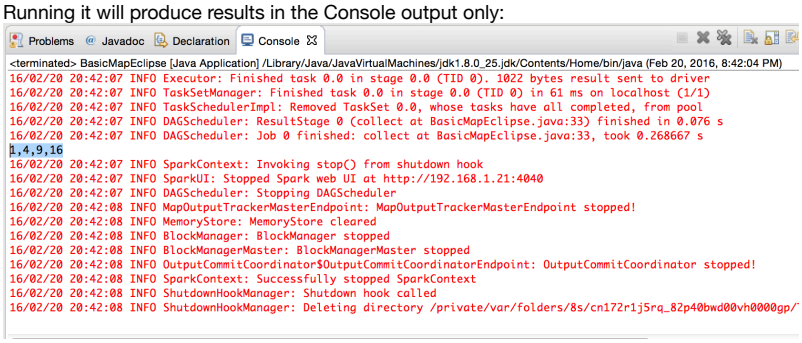
Refresh your Eclipse project (right-click on the Eclipse project name and click Refresh)
Inspect the results:



You can execute the BasicMapEclipse class in exactly the same way, except that it does not require any input or output parameters, so the Run Configuration is simpler, no Arguments are required:



Running it will produce results in the Console output only:



Creating JAR with Maven on Mac

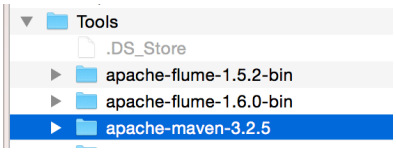
Pre-requisites:

- Java is installed

You can create your Spak job's JAR on your local machine - and later run on your VM.
I'm showing how it is done on Mac - it is very similar on Windows and *Nix systems as well.

If you do not have Maven - install it.
Simple download latest Maven binary from here: http://maven.apache.org/download.cgi

untar in some location - I got maven-3.2.5 on my Mac:



Make sure your JAVA_HOME is set correctly - Maven will not work without it being set.
Also, add Maven's bin directory to your PATH.
You can do this in your **.profile** :

> *MacBook-Pro-3:mini-examples-java marinapopova$ more ~/.profile*
> *export PATH=${PATH}:/Users/marinapopova/Marina/Tools/mongodb-2.6.6/bin:/Users/marinapopova/Marina/Tools/**apache-maven-3.2.5/bin***
> *export JAVA_HOME=`/usr/libexec/java_home`*

That's it. Now you can verify that you Maven works:

> *MacBook-Pro-3:mini-examples-java marinapopova$ **mvn -version***
> ***Apache Maven 3.2.5** (12a6b3acb947671f09b81f49094c53f426d8cea1; 2014-12-14T12:29:23-05:00)*
> *Maven home: /Users/marinapopova/Marina/Tools/apache-maven-3.2.5*
> *Java version: 1.8.0_25, vendor: Oracle Corporation*
> *Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/jre*

Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "10.10.5", arch: "x86_64", family: "mac"
Yottaas-MacBook-Pro-3:mini-examples-java marinapopova$

Now you can build your spark project with Maven.
Cd into your project's dir - where pom.xml is:

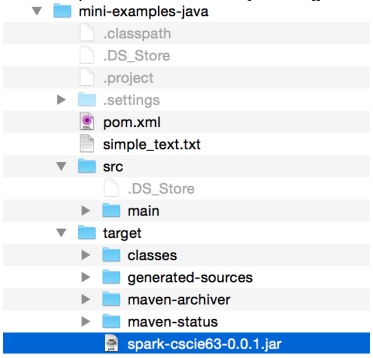
```
Yottaas-MacBook-Pro-3:mini-examples-java marinapopova$ pwd
/Users/marinapopova/Marina/GoogleDriveDir/Classes/CSCIE63_2016/Projects/mini-examples-java
Yottaas-MacBook-Pro-3:mini-examples-java marinapopova$ ls -R
pom.xml         simple_text.txt src

./src:
main

./src/main:
java

./src/main/java:
edu

./src/main/java/edu:
hu

./src/main/java/edu/hu:
examples

./src/main/java/edu/hu/examples:
BasicMap.java           BasicMapEclipse.java    WordCount.java          WordCountEclipse.java
Yottaas-MacBook-Pro-3:mini-examples-java marinapopova$
```

I changed my pom.xml - to use a slightly different jar name:

<artifactId>**spark-cscie63**</artifactId>

run: mvn clean package

Result: spark-cscie63-0.0.1.jar — generated in the target dir:



Now you can copy this jar to your VM and run it there via the normal "spark-submit" command - as shown below with the jars created on VM directly.

Compiling and building Spark Java jobs on CentOS6.7 VM

Pre-requisites:

- CDH5.5 is installed
- Maven is installed
- Spark is installed

Create a new directory for Spark projects - and copy pom.xml and all src/ content (code) that was provided.
I have copied all fields to my shared VM_shared folder for convenience - you can scp as well:

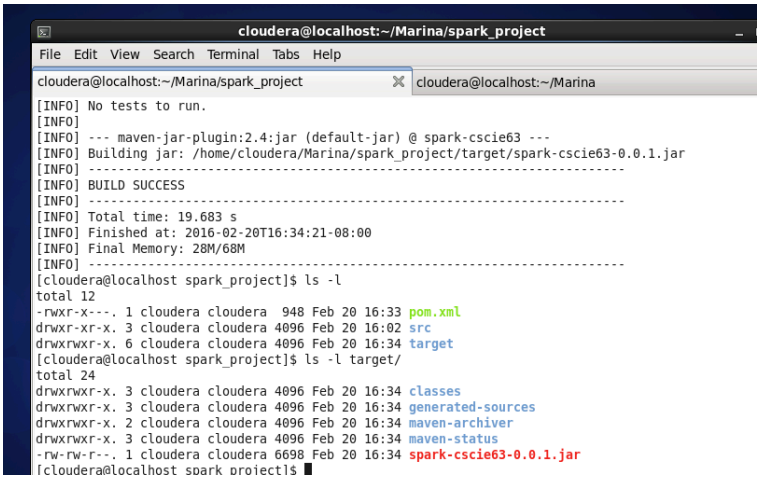
```
mkdir spark_project
cd spark_project/
cp -r /mnt/hgfs/VM_shared/spark_project/* .
```

```
[cloudera@localhost spark_project]$ pwd
/home/cloudera/Marina/spark_project
[cloudera@localhost spark_project]$ ls -R
.:
pom.xml  src

./src:
main

./src/main:
java

./src/main/java:
edu

./src/main/java/edu:
hu

./src/main/java/edu/hu:
examples

./src/main/java/edu/hu/examples:
BasicMap.java  WordCount.java
[cloudera@localhost spark_project]$
```

OPTIONAL: modify pom.xml to create a jar with a different name (not spark-examples, to avoid potential collision with the original Spark examples jar ….):

```
<project>
  <groupId>edu.hu.examples</groupId>
  <artifactId>spark-cscie63</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <name>cscie63_wordcount</name>
  <packaging>jar</packaging>
  <version>0.0.1</version>
  <dependencies>
    <dependency> <!-- Spark dependency -->
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.10</artifactId>
      <version>1.1.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
```

[cloudera@localhost spark_project]$ mvn clean package



prepare local input for word count job - I used my own small file to make sure I can verify results easily:

```
[cloudera@localhost Marina]$ more local_simple_text.txt
some sample input text
another line of text
more text on line
[cloudera@localhost Marina]$
```

[cloudera@localhost Marina]$ **$SPARK_HOME/bin/spark-submit --class edu.hu.examples.WordCount --master local[2] /home/cloudera/Marina/spark_project/target/spark-cscie63-0.0.1.jar file:///home/cloudera/Marina/local_simple_text.txt spark_output**

partial output - last few lines:

16/02/20 16:41:51 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/jobs/json,null}
16/02/20 16:41:51 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/jobs,null}
16/02/20 16:41:51 INFO ui.SparkUI: Stopped Spark web UI at http://192.168.177.182:4040
16/02/20 16:41:51 INFO scheduler.DAGScheduler: Stopping DAGScheduler
16/02/20 16:41:51 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/02/20 16:41:51 INFO storage.MemoryStore: MemoryStore cleared
16/02/20 16:41:51 INFO storage.BlockManager: BlockManager stopped
16/02/20 16:41:51 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
16/02/20 16:41:51 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/02/20 16:41:51 INFO spark.SparkContext: Successfully stopped SparkContext
16/02/20 16:41:51 INFO util.ShutdownHookManager: Shutdown hook called
16/02/20 16:41:51 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-af4b130f-1d2a-4506-8b56-a43d7d6ad50e
16/02/20 16:41:51 INFO remote.RemoteActorRefProvider$RemotingTerminator: Shutting down remote daemon.
16/02/20 16:41:51 INFO remote.RemoteActorRefProvider$RemotingTerminator: Remote daemon shut down; proceeding with flushing remote transports.
16/02/20 16:41:51 INFO Remoting: Remoting shut down
16/02/20 16:41:51 INFO remote.RemoteActorRefProvider$RemotingTerminator: Remoting shut down.
[cloudera@localhost Marina]$

Results are correct:

```
[cloudera@localhost Marina]$ hadoop fs -ls
Found 2 items
-rw-r--r--   1 cloudera supergroup         61 2016-02-20 13:11 simple_text.txt
drwxr-xr-x   - cloudera supergroup          0 2016-02-20 16:41 spark_output
[cloudera@localhost Marina]$ hadoop fs -ls spark_output
Found 3 items
-rw-r--r--   1 cloudera supergroup          0 2016-02-20 16:41 spark_output/_SUCCESS
-rw-r--r--   1 cloudera supergroup         39 2016-02-20 16:41 spark_output/part-00000
-rw-r--r--   1 cloudera supergroup         44 2016-02-20 16:41 spark_output/part-00001
[cloudera@localhost Marina]$ hadoop fs -cat spark_output/part-00000
(line,2)
(some,1)
(input,1)
(sample,1)
[cloudera@localhost Marina]$ hadoop fs -cat spark_output/part-00001
(another,1)
(text,3)
(on,1)
(more,1)
(of,1)
[cloudera@localhost Marina]$
```