Twitter Streaming with Spark

Spark Streaming allows us to analyze continuous streams of data. Eg:- Processing log data from websites or servers. This data is aggregated and analyzed at some given interval. We can feed this data to some port like Amazon Kinesis, HDFS, Kafka, Flume, etc. The 'checkpointing' feature stores data to disk periodically for fault tolerance.

Just like Spark has Dataframes and Datasets, Spark Streaming has 'Dstream' objects, which breaks up a stream of continuous data into distinct RDDs. AN advantage to this is that we can apply a lot of concepts of Dataframes to Dstreams as well.

Streamin can be done on a local port or on any particular app. One thing to keep in mind is that when streaming, our RDDs may contain very less data. For this we can increase the timeframe of our processing through 'windowed operations', which combine results from multiple batches over a sliding time window.

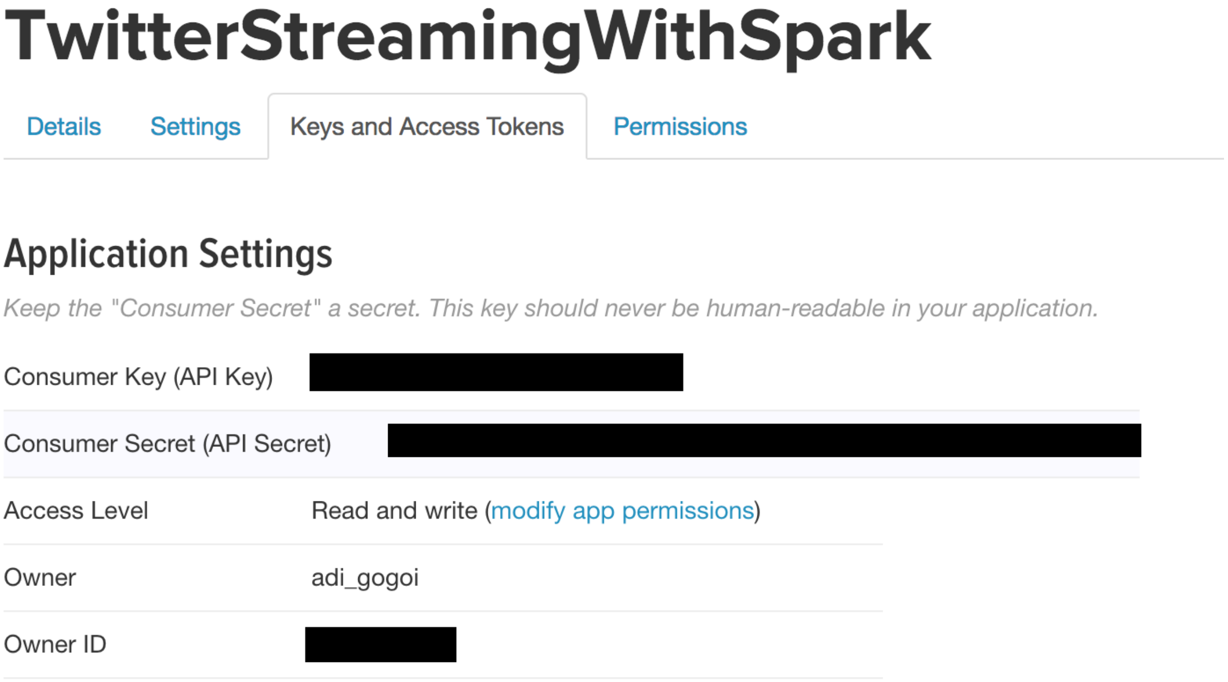
We can also keep track of an event across many batches through time by using 'updateStateByKey()' function.

In this project, we will stream Twitter and keep track of few of the most popular hashtags (#) as we go.

Creating a Twitter App:

In order to get data from Twitter, we need to create an app on twitter and accessing its data through our code. It involves the following;

1. Creating the App on Twitter. I have created an App called 'TwitterStreaminWithSpark'. With each app comes a Consumer (or API) Key and Secret. This will be useful for us while establishing connection with Twitter in our code.

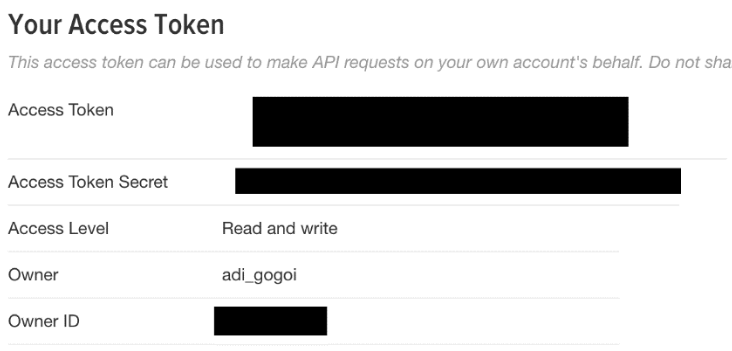


Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[redacted]
Consumer Secret (API Secret)	[redacted]
Access Level	Read and write (modify app permissions)
Owner	adi_gogoi
Owner ID	[redacted]

2. After creating an app, we need to create an Access Token for our app to function. This token consists of an Access Token and Secret. Keep this recorded for future use.

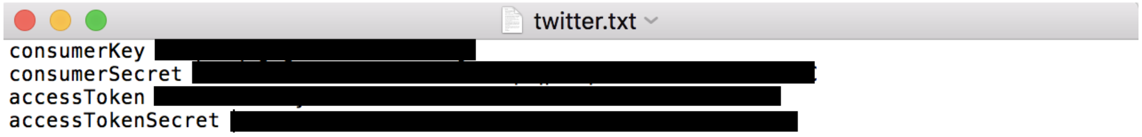


Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[redacted]
Access Token Secret	[redacted]
Access Level	Read and write
Owner	adi_gogoi
Owner ID	[redacted]

3. Next we need to create the configuration file for Twitter, which will consist of our Consumer and Access Tokens and Keys. This will be reference for our program when it will be accessing Twitter's Streaming data.

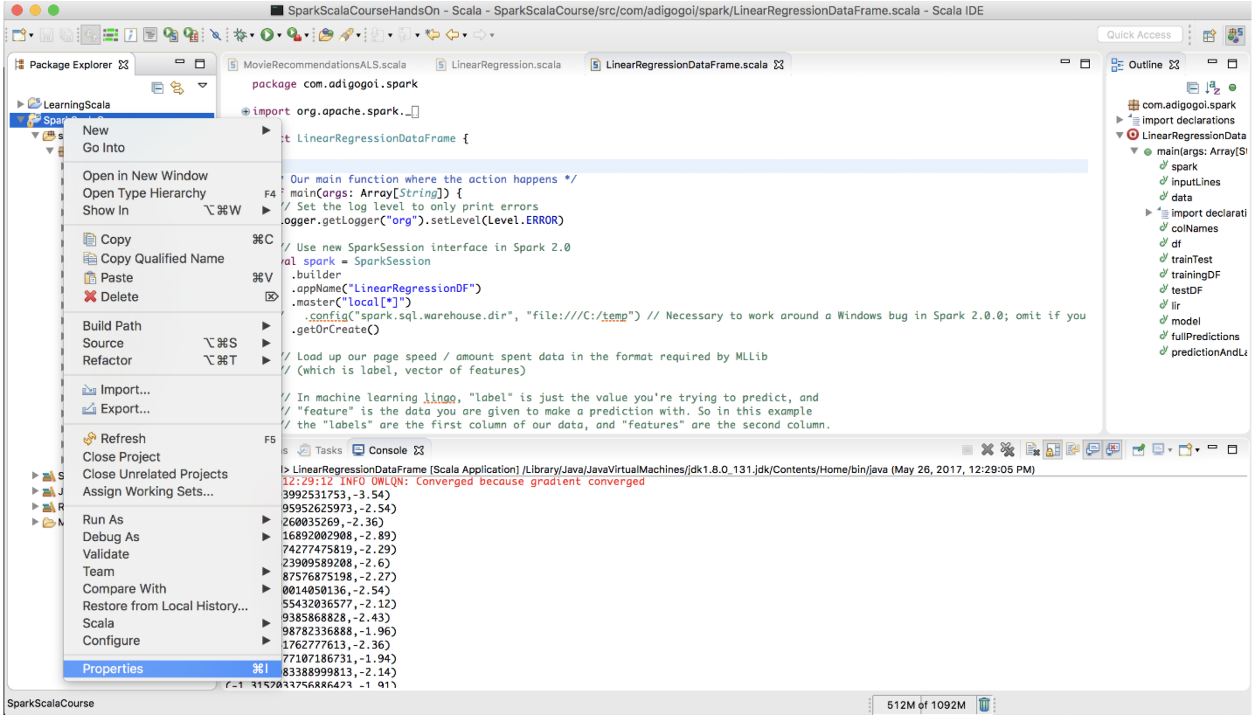


```
consumerKey [redacted]
consumerSecret [redacted]
accessToken [redacted]
accessTokenSecret [redacted]
```

Including Twitter Jars

Working with twitter is not possible until we include libraries that allow Spark to integrate with the Twitter API. This can be done by including external jars in our project.

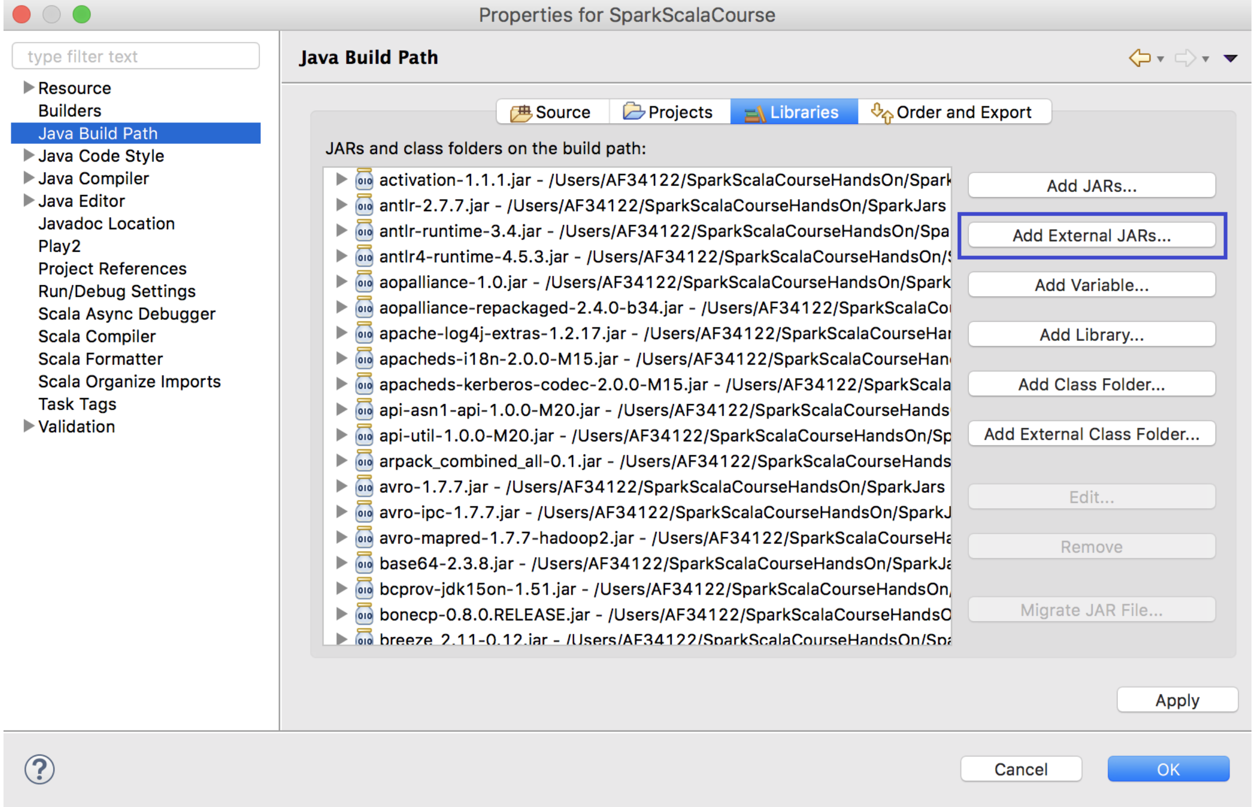
1. Go to the Properties of the Project folder.



```
package com.adigogoi.spark

import org.apache.spark._

:t LinearRegressionDataFrame {

 Our main function where the action happens */
 main(args: Array[String]) {
/ Set the log level to only print errors
.ogger.getLogger("org").setLevel(Level.ERROR)

/ Use new SparkSession interface in Spark 2.0
al spark = SparkSession
 .builder
 .appName("LinearRegressionDF")
 .master("local[*]")
   .config("spark.sql.warehouse.dir", "file:///C:/temp") // Necessary to work around a Windows bug in Spark 2.0.0; omit if you
 .getOrCreate()

/ Load up our page speed / amount spent data in the format required by MLLib
/ (which is label, vector of features)

/ In machine learning lingo, "label" is just the value you're trying to predict, and
/ "feature" is the data you are given to make a prediction with. So in this example
/ the "labels" are the first column of our data, and "features" are the second column.
```

```
> LinearRegressionDataFrame [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (May 26, 2017, 12:29:05 PM)
12:29:12 INFO OWLQN: Converged because gradient converged
3992531753,-3.54)
95952625973,-2.54)
260035269,-2.36)
16892002908,-2.89)
74277475819,-2.29)
23909589208,-2.6)
87576875198,-2.27)
0014050136,-2.54)
55432036577,-2.12)
9385868828,-2.43)
98782336888,-1.96)
1762777613,-2.36)
77107186731,-1.94)
83388999813,-2.14)
(-1.3152033756886423,-1.91)
```

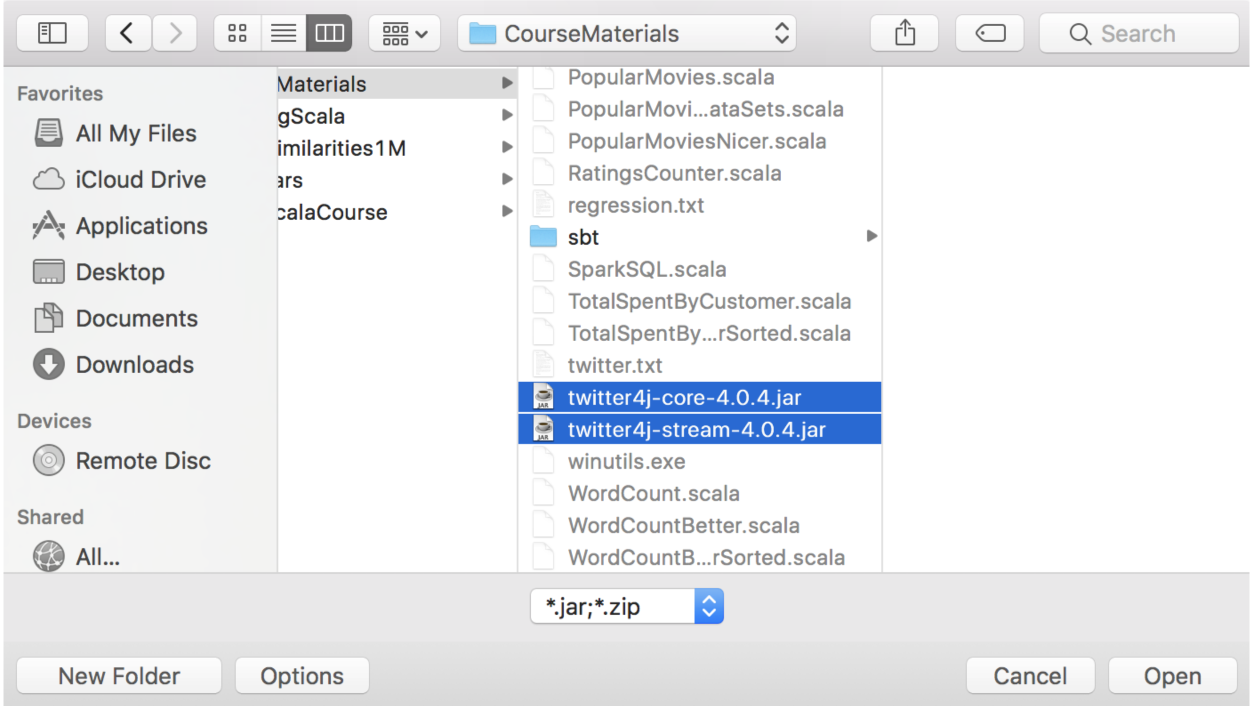
2. Got to ‘Java Build Path’ and then ‘Add External Jars’.



JARs and class folders on the build path:

activation-1.1.1.jar - /Users/AF34122/SparkScalaCourseHandsOn/Spark
antlr-2.7.7.jar - /Users/AF34122/SparkScalaCourseHandsOn/SparkJars
antlr-runtime-3.4.jar - /Users/AF34122/SparkScalaCourseHandsOn/Spa
antlr4-runtime-4.5.3.jar - /Users/AF34122/SparkScalaCourseHandsOn/S
aopalliance-1.0.jar - /Users/AF34122/SparkScalaCourseHandsOn/Spark
aopalliance-repackaged-2.4.0-b34.jar - /Users/AF34122/SparkScalaCo
apache-log4j-extras-1.2.17.jar - /Users/AF34122/SparkScalaCourseHan
apacheds-i18n-2.0.0-M15.jar - /Users/AF34122/SparkScalaCourseHan
apacheds-kerberos-codec-2.0.0-M15.jar - /Users/AF34122/SparkScala
api-asn1-api-1.0.0-M20.jar - /Users/AF34122/SparkScalaCourseHands
api-util-1.0.0-M20.jar - /Users/AF34122/SparkScalaCourseHandsOn/Sp
arpack_combined_all-0.1.jar - /Users/AF34122/SparkScalaCourseHands
avro-1.7.7.jar - /Users/AF34122/SparkScalaCourseHandsOn/SparkJars
avro-ipc-1.7.7.jar - /Users/AF34122/SparkScalaCourseHandsOn/SparkJ
avro-mapred-1.7.7-hadoop2.jar - /Users/AF34122/SparkScalaCourseHa
base64-2.3.8.jar - /Users/AF34122/SparkScalaCourseHandsOn/SparkJa
bcprov-jdk15on-1.51.jar - /Users/AF34122/SparkScalaCourseHandsOn,
bonecp-0.8.0.RELEASE.jar - /Users/AF34122/SparkScalaCourseHandsO
breeze_2.11-0.12.jar - /Users/AF34122/SparkScalaCourseHandsOn/Spa

3. Select the three JARs necessary for Twitter Streaming i.e. dstream-twitter_2.11-0.1.0-SNAPSHOT, twitter4j-core-4.0.4, and twitter4j-stream-4.0.4 and click Open.



```
Materials
gScala
imilarities1M
ars
calaCourse
```

```
PopularMovies.scala
PopularMovi...ataSets.scala
PopularMoviesNicer.scala
RatingsCounter.scala
regression.txt
sbt
SparkSQL.scala
TotalSpentByCustomer.scala
TotalSpentBy...rSorted.scala
twitter.txt
twitter4j-core-4.0.4.jar
twitter4j-stream-4.0.4.jar
winutils.exe
WordCount.scala
WordCountBetter.scala
WordCountB...rSorted.scala
```

.jar;.zip

Twitter Streaming

After including the necessary jars, we can Stream Twitter data to find the most popular Hashtags. But before that, we should know how our code works:

1. We use the createStream function of TwitterUtils to create the Dstream and store it in ‘tweets’ variable.

2. We then map the status of the tweets in this Dstream and get them stored in the variable ‘statuses’.

3. Using flatMap, we split the tweets into words and store it in ‘tweetwords’.

4. We then use filter to get just those words that begin with ‘#’ and store it in the variable ‘hashtags’.

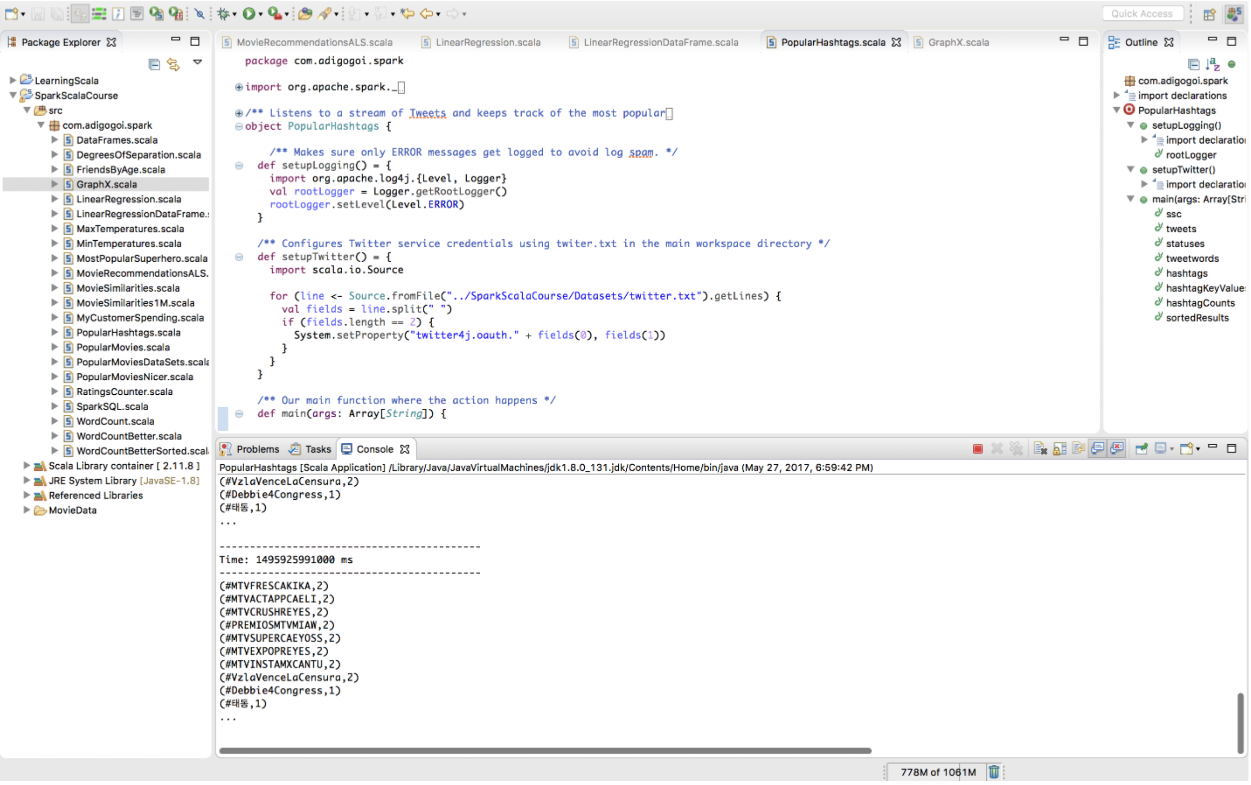
5. Then we can convert these hashtags into key-value pairs, with the hashtags being the key and their count being the value. This involves first mapping each hashtag with 1 and then reduce with count.

6. Next we sort the key-value pairs in descending order and display them.

Another major part of the code is the ‘setuptwitter’ function, which reads the configuration file for twitter, gets the keys and sets up the System for Twitter Authentication.

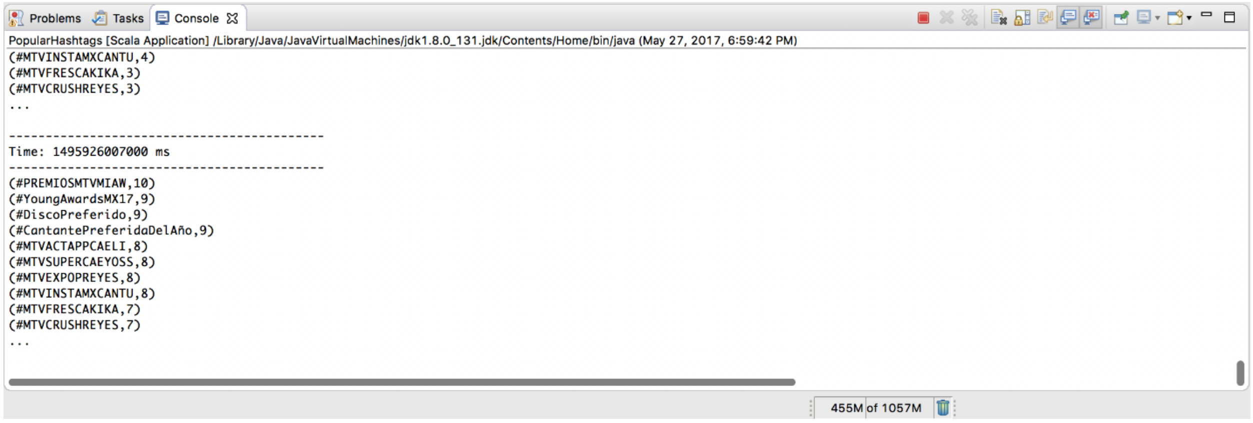
We also need to note that unlike the usual SparkContext, we need to use the StreamingContext for this project. We also need to setup a checkpoint folder before starting the context for Spark Streaming.

Let’s check the first few screenshots of the most popular hashtags of our 5 second window.

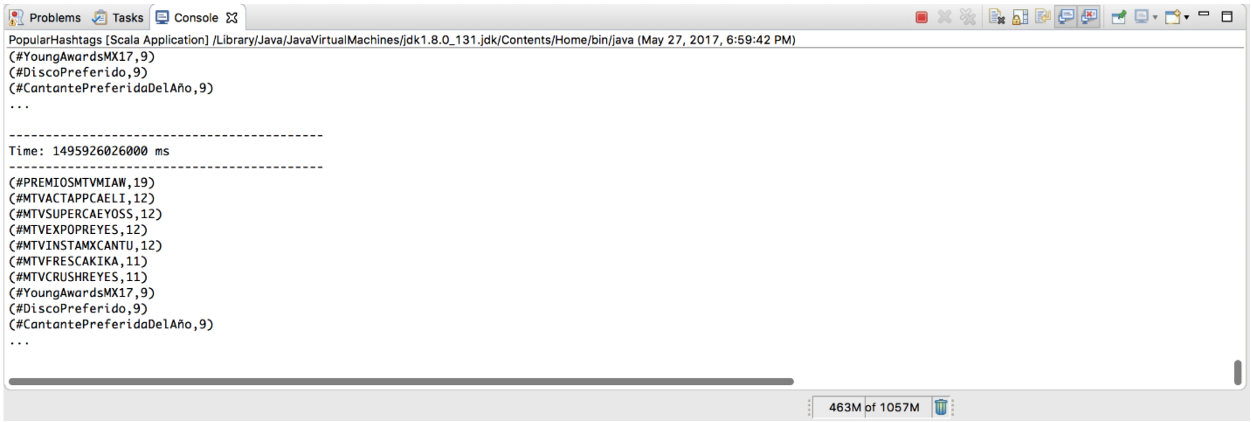


```
package com.adigogoi.spark

import org.apache.spark._

/** Listens to a stream of Tweets and keeps track of the most popular
object PopularHashtags {

    /** Makes sure only ERROR messages get logged to avoid log spam. */
  def setupLogging() = {
    import org.apache.log4j.{Level, Logger}
    val rootLogger = Logger.getRootLogger()
    rootLogger.setLevel(Level.ERROR)
  }

  /** Configures Twitter service credentials using twiter.txt in the main workspace directory */
  def setupTwitter() = {
    import scala.io.Source

    for (line <- Source.fromFile("../SparkScalaCourse/Datasets/twitter.txt").getLines) {
      val fields = line.split(" ")
      if (fields.length == 2) {
        System.setProperty("twitter4j.oauth." + fields(0), fields(1))
      }
    }
  }

  /** Our main function where the action happens */
  def main(args: Array[String]) {
```

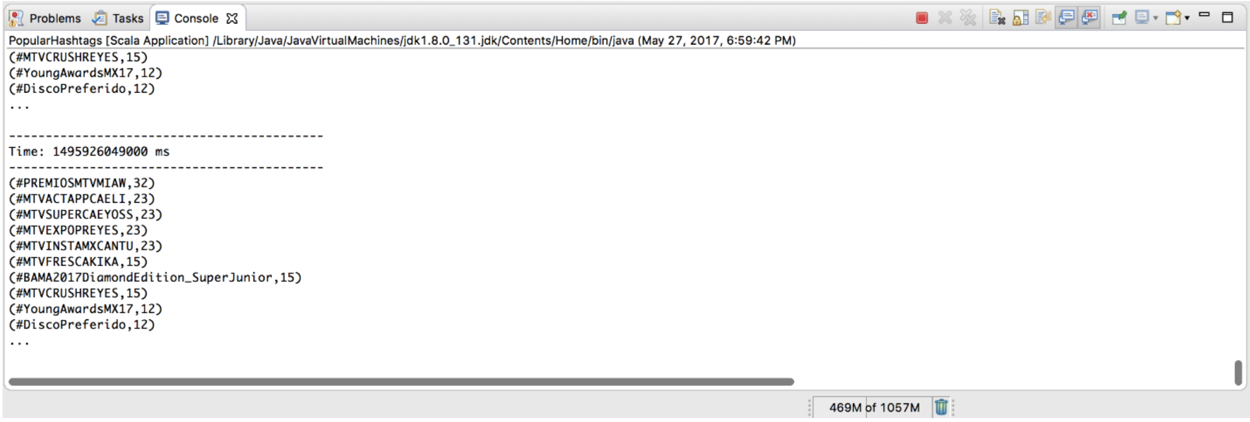
```
PopularHashtags [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (May 27, 2017, 6:59:42 PM)
(#VzlaVenceLaCensura,2)
(#Debbie4Congress,1)
(#태동,1)
...

-------------------------------------------
Time: 1495925991000 ms
-------------------------------------------
(#MTVFRESCAKIKA,2)
(#MTVACTAPPCAELI,2)
(#MTVCRUSHREYES,2)
(#PREMIOSMTVMIAW,2)
(#MTVSUPERCAEYOSS,2)
(#MTVEXPOPREYES,2)
(#MTVINSTAMXCANTU,2)
(#VzlaVenceLaCensura,2)
(#Debbie4Congress,1)
(#태동,1)
...
```



```
PopularHashtags [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (May 27, 2017, 6:59:42 PM)
(#MTVINSTAMXCANTU,4)
(#MTVFRESCAKIKA,3)
(#MTVCRUSHREYES,3)
...

-------------------------------------------
Time: 1495926007000 ms
-------------------------------------------
(#PREMIOSMTVMIAW,10)
(#YoungAwardsMX17,9)
(#DiscoPreferido,9)
(#CantantePreferidaDelAño,9)
(#MTVACTAPPCAELI,8)
(#MTVSUPERCAEYOSS,8)
(#MTVEXPOPREYES,8)
(#MTVINSTAMXCANTU,8)
(#MTVFRESCAKIKA,7)
(#MTVCRUSHREYES,7)
...
```



```
PopularHashtags [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (May 27, 2017, 6:59:42 PM)
(#YoungAwardsMX17,9)
(#DiscoPreferido,9)
(#CantantePreferidaDelAño,9)
...

-------------------------------------------
Time: 1495926026000 ms
-------------------------------------------
(#PREMIOSMTVMIAW,19)
(#MTVACTAPPCAELI,12)
(#MTVSUPERCAEYOSS,12)
(#MTVEXPOPREYES,12)
(#MTVINSTAMXCANTU,12)
(#MTVFRESCAKIKA,11)
(#MTVCRUSHREYES,11)
(#YoungAwardsMX17,9)
(#DiscoPreferido,9)
(#CantantePreferidaDelAño,9)
...
```



```
PopularHashtags [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (May 27, 2017, 6:59:42 PM)
(#MTVCRUSHREYES,15)
(#YoungAwardsMX17,12)
(#DiscoPreferido,12)
...

-------------------------------------------
Time: 1495926049000 ms
-------------------------------------------
(#PREMIOSMTVMIAW,32)
(#MTVACTAPPCAELI,23)
(#MTVSUPERCAEYOSS,23)
(#MTVEXPOPREYES,23)
(#MTVINSTAMXCANTU,23)
(#MTVFRESCAKIKA,15)
(#BAMA2017DiamondEdition_SuperJunior,15)
(#MTVCRUSHREYES,15)
(#YoungAwardsMX17,12)
(#DiscoPreferido,12)
...
```

We can see from the output of Twitter Streaming that some hashtags of MTV are trending quite a lot now.

This project helped me realize that Spark Streaming is a great tool not just to store streaming data, but also to perform real-time analytics on the data and give insights.