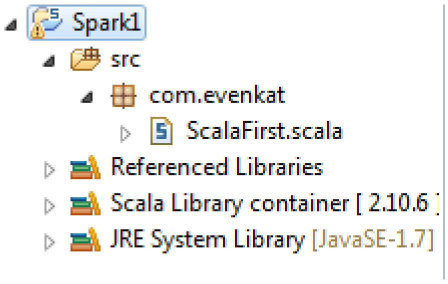
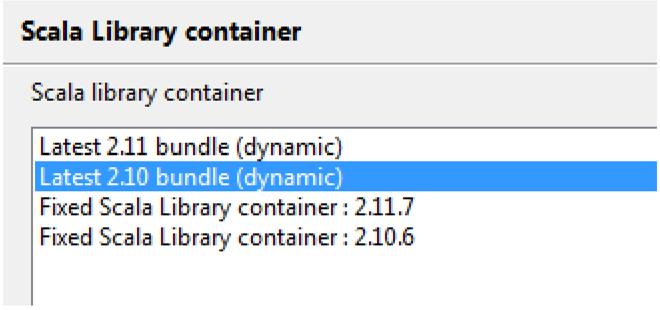
1) Create a Scala Project called Spark1 and inside that create a package com.evenkat and within this package define a new Scala object by the name of ScalaFirst.



```
◢ Spark1
    ◢ src
        ◢ com.evenkat
            ▷ ScalaFirst.scala
    ▷ Referenced Libraries
    ▷ Scala Library container [ 2.10.6 ]
    ▷ JRE System Library [JavaSE-1.7]
```

VERY IMP: The Scala Library Container should be 2.10 and not 2.11. For checking / changing this right click on the Scala Library Container and go to properties and ensure that the selection is 2.10 bundle (dynamic)



Scala Library container

Scala library container

- Latest 2.11 bundle (dynamic)
- [✓ Latest 2.10 bundle (dynamic)]
- Fixed Scala Library container : 2.11.7
- Fixed Scala Library container : 2.10.6

2) Define a main method in the Scala object and also import SparkConf and SparkContext:

```
package com.evenkat

import org.apache.spark.{SparkConf, SparkContext}

object ScalaFirst {

 def main(args: Array[String]){
  //Scala Main Method
 }
}
```

3) Import the spark-assembly-1.6.0-hadoop2.6.0.jar with Add Additional Jar and ensure in the Order and Import Tab that this jar comes before the Scala Library Containr.

4) Add the following lines inside the main function.

```
println("Creating Spark Configuration")
//Create an Object of Spark Configuration
val conf = new SparkConf()
//Set the logical and user defined Name of this Application
conf.setAppName("My First Spark Scala Application")
```

```
    //Define the URL of the Spark Master.
    //Useful only if you are executing Scala App directly //from the console.
    //We will comment it for now but will use later
    //conf.setMaster("spark://ip-10-237-224-94:7077")

    println("Creating Spark Context")
    //Create a Spark Context and provide previously created
    //Object of SparkConf as an reference.
    val ctx = new SparkContext(conf)

    println("Loading the Dataset and will further process it")

    //Defining and Loading the Text file from the local //file system or HDFS
    //and converting it into RDD.
    //SparkContext.textFile(..) - It uses the Hadoop's //TextInputFormat and file
is
    //broken by New line Character.
    //Refer to
http://hadoop.apache.org/docs/r2.6.0/api/org/apache/hadoop/mapred/TextInputFormat.
html
    //The Second Argument is the Partitions which specify //the parallelism.
    //It should be equal or more then number of Cores in //the cluster.
    val file = "/input/sample"
    val logData = ctx.textFile(file, 2)

    //Invoking Filter operation on the RDD.
    //And counting the number of lines in the Data loaded //in RDD.
    //Simply returning true as "TextInputFormat" have //already divided the data
by "\n"
    //So each RDD will have only 1 line.
    val numLines = logData.filter(line => true).count()

    //Finally Printing the Number of lines.
println("Number of Lines in the Dataset " + numLines)
```

5) Ensure that you copy sample to the / location of HDFS via the command.

```
hdfs dfs –put LocationofSample /
```

6) Create a jar file and move it to Ubuntu where the SPARK_HOME is present [typically in /home/notroot/lab/software/spark-1.6.0-bin-hadoop2.6/]

7) Execute it using spark-submit which we used and for this example, we are not using any input of output parameters since we are printing within the code itself. Run this from the SPARK_HOME/bin location:

```
spark-submit --class com.evenkat.ScalaFirst --master local ../ScalaFirst.jar
```