// ( a )

. flatMap ( l -> Arrays . asList ( l . split ( " " ) ) . iterator ( ) )

FlatMap wendet die darinstehende Lambda Funktion auf alle Strings im RDD an. Die durch die Lambda Funktion entstehenden Ergebnislisten werden konkateniert und als neue RDD zurückgegeben.

Die Lambda Funktion selbst nimmt einen Parameter l entgegen. Dieser wird zunächst an den Leerzeilen gesplittet so dass ein String Array entsteht. Dieses Array wird als Liste konvertiert und dessen Iterator an die flatMap Methode zurückgegeben.

So ist die Rückgabe JavaRDD<String>

// (b)

. mapToPair ( w -> new Tuple2<String , Integer>(w , 1) )

Die mapToPair Methode wendet die darinstehende Lambda Funktion auf alle in ihr stehenden Strings im RDD an.

Die Lambda Funktion hat einen Eingabeparameter w und bildet aus diesem ein Schlüssel werte Paar durch die Verwendung von Tuple2. Das Wort w ist der Schlüssel und erhält eine eins als wert. mapToPair liefert dann als Rückgabe JavaPairRDD<String,Integer>.

// ( c )

. reduceByKey ( ( x , y ) -> x + y )

Die reduceByKey Funktion gruppiert die Pairs im RDD nach Schlüssel und wendet dann die in ihr stehende Lambda Funktion an so dass jeder Key nur noch ein Mal im RDD vorhanden ist. Diese Funktion erhält als Eingabe ein Tupel welche die Wert von zwei Pairs mit gleichem Schlüssel darstellen, also in unserem Fall zwei Integer Werte. Diese Werte werden innerhalb der Lambda Funktion aufsummiert. reduceByKey liefert dann als Rückgabe JavaPairRDD<String,Integer>. Wobei in jedem Pair ein Wort als Schlüssel steht und dessen Häufigkeit als wert.

|  |
| --- |
|  |