===SharedVariables

**import** org.apache.spark.broadcast.Broadcast  
**import** org.apache.spark.rdd.RDD  
**import** org.apache.spark.{SparkConf, SparkContext}  
  
*/\*\*  
 \* Created by Administrator on 2017/3/19 0019.  
 \*/***object** SharedVariables {  
 **def** main(args: Array[String]): Unit = {  
 **val** conf = **new** SparkConf().setAppName(**"SharedVariables"**).setMaster(**"local[2]"**)  
 **val** sc = **new** SparkContext(conf)  
  
 *//广播变量* **val** broadcastVar = sc.broadcast(3)  
 *println*(broadcastVar.value)  
 *//初始化RDD* **var** rdd: RDD[Int] = sc.parallelize(*Array*(1,2,3))  
 rdd.map(x => x + broadcastVar.value).foreach(*println*(\_))  
  
 *//(userId,name,clickTime)* **val** clicks =*Array*((1001,**"张三"**,123456789),(1002,**"李四"**,123456789),(1003,**"王五"**,123456789))  
 **var** clicksRdd: RDD[(Int, String, Int)] = sc.parallelize(clicks)  
 **var** broadcastRdd: Broadcast[RDD[(Int, String, Int)]] = sc.broadcast(clicksRdd)  
  
 *//(userId,name,isBlack)* **val** blacks = *Array*((1001,**"张三"**,0),(1002,**"李四"**,1),(1004,**"赵六"**,1),(1005,**"孙七"**,1))  
 **var** blacksRdd: RDD[(Int, String, Int)] = sc.parallelize(blacks)  
  
 *//数据转换* **var** clickBlackRdd: RDD[(Int, (String, Option[Int]))] = broadcastRdd.value.map(x => (x.\_1,x.\_2)).leftOuterJoin( blacksRdd.map(x => (x.\_1,x.\_3)) )  
 clickBlackRdd.foreach(*println*(\_))  
  
 clickBlackRdd.filter( x => x.\_2.\_2.nonEmpty && x.\_2.\_2.get == 1).foreach(x => *println*(**"点击用户存在黑明单的是："** + x))  
  
 Thread.*sleep*(60000)  
 sc.stop()  
 }  
}

====Accumulators

**import** org.apache.spark.{Accumulable, AccumulatorParam, SparkConf, SparkContext}  
  
*/\*\*  
 \* Created by Administrator on 2017/3/19 0019.  
 \*/***object** VectorAccumulatorParam **extends** AccumulatorParam[Long] {  
 *//初始化* **def** zero(initialValue: Long): Long = {  
 0  
 }  
 *//累加方法* **def** addInPlace(v1: Long, v2: Long): Long = {  
 v1 + v2  
 }  
}  
  
**object** Accumulators {  
 **def** main(args: Array[String]): Unit = {  
 **val** conf = **new** SparkConf().setAppName(**"Accumulators"**).setMaster(**"local[4]"**)  
 **val** sc = **new** SparkContext(conf)  
  
*// val accum = sc.accumulator(0, "Accumulator")  
// sc.parallelize(Array(1, 2, 3, 4)).foreach(x => accum += x)  
// println(accum.value)  
  
 //自定义累加器* **var** accumulable: Accumulable[Long, Long] = sc.accumulable(0l,**"My Accumulator"**)(VectorAccumulatorParam)  
 sc.parallelize(*Array*(1, 2, 3, 4)).foreach(x => accumulable += x)  
 *println*(accumulable.value)  
  
 Thread.*sleep*(60000)  
 sc.stop()  
 }  
}