**val** baseData = kafkardd.map(t=>JSON.*parseObject*(t.\_2))  
 .filter(\_.getString("serviceName").equalsIgnoreCase("reChargeNotifyReq"))  
 .map(jsobj=>{  
 **val** result = jsobj.getString("bussinessRst") //充值结果  
 **val** fee:Double = **if**(result.equals("0000"))  
 jsobj.getDouble("chargefee") **else** 0.0 //充值金额  
 **val** feeCount = **if**(!fee.equals(0.0)) 1 **else** 0 //获取到充值成功数  
 **val** starttime = jsobj.getString("requestId") //开始充值时间  
 **val** recivcetime = jsobj.getString("receiveNotifyTime") //结束充值时间  
 **val** pcode = jsobj.getString("provinceCode") //获得省份编号  
 **val** city = broadcasts.value.get(pcode).toString //通过省份编号进行取值  
 //充值成功数  
 **val** isSucc = **if**(result.equals("0000")) 1 **else** 0  
 // 充值时长  
 **val** costtime = **if**(result.equals("0000")) Utils.*costtime*(starttime,recivcetime) **else** 0  
 (starttime.substring(0,8),  
 starttime.substring(0,10),  
 *List*[Double](1,fee,isSucc,costtime.toDouble,feeCount),  
 city,  
 starttime.substring(0,12),  
 (starttime.substring(0,10),city)  
 )  
 //持久化RDD  
 }).cache()  
//处理第一个指标  
//要将两个list拉倒一起去，因为每次处理的结果要合并  
**val** result = baseData.map(t=>(t.\_1,t.\_3)).reduceByKey((list1,list2)=>{  
 //拉链操作  
 // 指标 一。  
 JedisApp.*Jedis01*(result)  
 list1.zip(list2).map(t=>t.\_1+t.\_2)  
})  
 // 第二个指标分析  
**val** result2 = baseData.map(t=>(t.\_6,t.\_3)).reduceByKey((list1,list2)=>{  
 list1.zip(list2).map(t=>t.\_1+t.\_2)  
})  
JedisApp.*Jedis02*(result2)  
  
// 指标统计三  
**val** result3 = baseData.map(t=>(t.\_4,t.\_3)).reduceByKey((list1,list2)=>{  
 //拉链操作  
 list1.zip(list2).map(t=>t.\_1+t.\_2)  
})  
JedisApp.*Jedis03*(result3)  
//指标四  
//要将两个list拉倒一起去，因为每次处理的结果要合并  
**val** result4 = baseData.map(t=>(t.\_5,t.\_3)).reduceByKey((list1,list2)=>{  
 //拉链操作  
 list1.zip(list2).map(t=>t.\_1+t.\_2)  
})  
JedisApp.*Jedis04*(result4)

*/\*\*  
 \* 第一个指标  
 \** ***@param lines*** *\*/* **def** Jedis01(lines :RDD[(String, List[Double])]): Unit = {  
 lines.foreachPartition(f=>{  
 **val** jedis = JedisConnectionPool.*getConntection*()  
 f.foreach(t=>{  
 //充值总单数  
 jedis.hincrBy(t.\_1,"total",t.\_2(0).toLong)  
 //充值总金额  
 jedis.hincrByFloat(t.\_1,"money",t.\_2(1))  
 //充值成功数  
 jedis.hincrBy(t.\_1,"success",t.\_2(2).toLong)  
 //充值总时长  
 jedis.hincrBy(t.\_1,"time",t.\_2(3).toLong)  
 })  
 jedis.close()  
 })  
 }  
  
 */\*\*  
 \* 指标二  
 \** ***@param lines*** *\*/* **def** Jedis02(lines :RDD[((String, String), List[Double])]): Unit ={  
 lines.foreachPartition(f=>{  
 **val** jedis = JedisConnectionPool.*getConntection*()  
 f.foreach(f=>{  
 jedis.hincrBy(f.\_1.\_1,f.\_1.\_2,f.\_2.head.toLong-f.\_2(2).toLong)  
 })  
 jedis.close()  
 })  
 }  
  
 */\*\*  
 \* 指标三  
 \*/* **def** Jedis03 (lines :RDD[(String, List[Double])]): Unit ={  
 lines.sortBy(\_.\_2.head,**false**)  
 .map(t=>(t.\_1,(t.\_2(2)/t.\_2.head\*100).formatted("%.1f")))  
 .foreachPartition(t=>{  
 // 拿到连接  
 **val** conn = JdbcMysql.*getConn*()  
 t.foreach(t=>{  
 **val** sql = "insert into CityTopN(city,success) " +  
 "values('"+t.\_1.toString.replace("Some(","").replace(")","")+"',"+t.\_2.toDouble+")"  
 **val** state = conn.createStatement()  
 state.executeUpdate(sql)  
 })  
 JdbcMysql.*releaseCon*(conn)  
 })  
 }  
  
 */\*\*  
 \* 指标四  
 \** ***@param lines*** *\*/* **def** Jedis04(lines :RDD[(String, List[Double])]): Unit ={  
 lines.map(t=> {  
 CountMoney.*Cases*(*List*(t.\_1, t.\_2(1), t.\_2(4)))  
 }).map(t=>(t.map(\_.\_1),t.map(t=>(t.\_2,t.\_3))))  
 .reduceByKey((list1,list2)=>list1.zip(list2)  
 .map(t=>(t.\_1.\_1+t.\_2.\_1,t.\_1.\_2+t.\_2.\_2)))  
 .foreachPartition(t=>{  
 // 拿到连接  
 **val** conn = JdbcMysql.*getConn*()  
 t.foreach(t=>{  
 **val** sql = "insert into RealTime(hour,count,money) " +  
 "values('"+t.\_1.toString().replace("List(","").replace(")","")+"'" +  
 ",'"+t.\_2.map(\_.\_1.toInt).toString().replace("List(","").replace(")","").toInt+"'" +  
 ",'"+t.\_2.map(\_.\_2.toDouble).toString().replace("List(","").replace(")","")+"')"  
 **val** state = conn.createStatement()  
 state.executeUpdate(sql)  
 })  
 JdbcMysql.*releaseCon*(conn)  
 })  
 }  
  
 */\*\*  
 \* 指标五  
 \** ***@param lines*** *\*/* **def** Jedis05(lines:RDD[((String, String), List[Double])],Dir:String): Unit ={  
 lines  
 // 设置时间日期  
 .filter(\_.\_1.\_1==Dir).map(t=>{  
 (t.\_1,t.\_2(0)-t.\_2(1),((t.\_2(0)-t.\_2(1))/t.\_2(0)\*100).formatted("%.2f"))  
 }).sortBy(\_.\_2,**false**).foreachPartition(t=>{  
 // 拿到连接  
 **val** conn = JdbcMysql.*getConn*()  
 t.foreach(t=>{  
 **val** sql = "insert into FailTopN(time,city,failnumber,failrate) " +  
 "values('"+t.\_1.\_1+"','"+t.\_1.\_2+"',"+t.\_2.toInt+","+t.\_3.toDouble+")"  
 **val** state = conn.createStatement()  
 state.executeUpdate(sql)  
 })  
 JdbcMysql.*releaseCon*(conn)  
 })  
 }  
}

**object** CityDay {  
 **def** main(args: Array[String]): Unit = {  
 System.*setProperty*("hadoop.home.dir", "D:\\Huohu\\下载\\hadoop-common-2.2.0-bin-master")  
 // 首先判断目录是否正确  
 **if**(args.length!=2){  
 *println*("退出程序")  
 sys.*exit*()  
 }  
 **val** *Array*(inputPath,startDir) = args  
 // 创建执行入口  
 **val** session = SparkSession.*builder*()  
 .appName("CityDay").master("local").getOrCreate()  
 // 获取到数据  
 **val** sc = session.sparkContext  
 **val** file = sc.textFile(inputPath)  
 // 广播变量  
 **val** test =sc  
 .textFile("D:\\资料\\Null\\spark\\充值平台实时统计分析\\city.txt")  
 .collect().map(t=>{  
 **val** l = t.split(" ")  
 (l(0),l(1))  
 }).toMap  
 **val** broadcasts= sc.broadcast(test)  
 // 处理业务  
 **val** lines = file.map(t=>{  
 **val** jsobj = JSON.*parseObject*(t)  
 **val** result = jsobj.getString("bussinessRst") //充值结果  
 **val** fee:Double = **if**(result.equals("0000"))  
 jsobj.getDouble("chargefee") **else** 0.0 //充值金额  
 **val** feeCount = **if**(!fee.equals(0.0)) 1 **else** 0 //获取到充值成功数  
 **val** starttime = jsobj.getString("requestId") //开始充值时间  
 **val** pcode = jsobj.getString("provinceCode") //获得省份编号  
 **val** city = broadcasts.value.getOrElse(pcode,"未知").toString //通过省份编号进行取值  
 //充值成功数  
 **val** isSucc = **if**(result.equals("0000")) 1 **else** 0  
 ((starttime.substring(0,8),city),  
 *List*[Double](1,isSucc,feeCount)  
 )  
 }).reduceByKey((list1,list2)=>{  
 list1.zip(list2).map(t=>t.\_1+t.\_2)  
 })  
 JedisApp.*Jedis05*(lines,startDir)  
 }  
}