1. 两种方式将RDD转换成SchemaRDD
   1. 通过定义case class，使用反射推断Schema（case class方式）；（使用前提：已经知道元数据信息）
   2. 通过可编程接口，定义Schema，并应用到RDD上（applySchema方式）；
2. 通过第一种方式case class
   1. Java

package com.java.SparkSQL;  
  
import java.io.Serializable;  
  
*/\*\*  
 \* Created by Administrator on 2016/3/17.  
 \*/*public class Person implements Serializable{  
  
 private int id;  
 private String name;  
 private int age;  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 @Override  
 public String toString() {  
 return "Person{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", age=" + age +  
 '}';  
 }  
}

package com.java.SparkSQL;  
  
import org.apache.spark.SparkConf;  
import org.apache.spark.api.java.JavaRDD;  
import org.apache.spark.api.java.JavaSparkContext;  
import org.apache.spark.api.java.function.Function;  
import org.apache.spark.sql.DataFrame;  
import org.apache.spark.sql.Row;  
import org.apache.spark.sql.SQLContext;  
  
import java.util.List;  
  
*/\*\*  
 \* Created by Administrator on 2016/3/17.  
 \*/*public class RDD2DataFrameByReflection {  
 public static void main(String[] args) {  
  
 SparkConf conf = new SparkConf().setAppName("RDD2DataFrameByFlection").setMaster("local");  
 JavaSparkContext sc = new JavaSparkContext(conf);  
 SQLContext sqlContext = new SQLContext(sc);  
  
 JavaRDD<String> lines = sc.textFile("E:/softwares/spark-1.6.0/examples/src/main/resources/person.txt");  
 JavaRDD<Person> persons = lines.map(new Function<String, Person>() {  
 @Override  
 public Person call(String line) throws Exception {  
 String[] splited = line.split("\t");  
 Person p = new Person();  
 p.setId(Integer.*valueOf*(splited[0]));  
 p.setName(splited[1]);  
 p.setAge(Integer.*valueOf*(splited[2]));  
 return p;  
 }  
 });  
 //RDD转换成DataFrame  
 //在底层通过反射的方式获得Person的所有fields，结合RDD本身，就生成了DataFrame  
 DataFrame df = sqlContext.createDataFrame(persons, Person.class);//第一个是RDD，第二个是对RDD的描述。  
 df.show();  
  
 df.registerTempTable("Person");  
 DataFrame bigData = sqlContext.sql("select \* from Person");  
   
 //DataFrame转化成RDD  
 JavaRDD<Row> bigDataRDD = bigData.javaRDD();  
 JavaRDD<Person> result = bigDataRDD.map(new Function<Row, Person>() {  
 @Override  
 public Person call(Row row) throws Exception {  
 Person p = new Person();  
 p.setId(row.getAs("id"));  
 p.setName(row.getAs("name"));  
 p.setAge(row.getAs("age"));  
 return p;  
 }  
 });  
 List<Person> personList = result.collect();  
 for(Person p : personList) {  
 System.*out*.println(p.toString());  
 }  
  
  
 }  
}

* 1. Scala

**package** scala.SparkSQL  
  
**import** org.apache.spark.sql.SQLContext  
**import** org.apache.spark.{SparkContext, SparkConf}  
  
*/\*\*  
 \* Created by Administrator on 2016/3/17.  
 \*/***case class** Person( id:Int, name:String, age:Int){  
 **override def** toString: String = (id+":"+name+":"+age)  
}  
**object** RDD2DataFrameByReflection {  
  
 **def** main(args:Array[String]): Unit = {  
  
 **val** conf = **new** SparkConf().setAppName("RDD2DataFrameFlection").setMaster("local")  
 **val** sc = **new** SparkContext(conf)  
 **val** sqlContext = **new** SQLContext(sc)  
  
 **val** lines = sc.textFile("E:/softwares/spark-1.6.0/examples/src/main/resources/person.txt")  
 **val** persons = lines.map(\_.split("\t"))  
 **import** sqlContext.implicits.\_  
 **val** df = persons.map(p => *Person*(p(0).toInt,p(1),p(2).toInt)).toDF();  
 df.show();  
  
 df.registerTempTable("person")  
 **val** bigData = sqlContext.sql("select \* from person")  
 //DataFrame转化成RDD  
 **val** bigDataRDD = bigData.*rdd* bigDataRDD.map(x=>**new** Person(x.getAs("id"),x.getAs("name"),x.getAs("age"))).foreach(x=>*println*(x.name))  
 bigDataRDD.map(x=>(x.getAs("id"),x.getAs("name"),x.getAs("age"))).foreach(x=>*println*(x.\_1.toString +""+x.\_2.toString))  
 **val** result = bigDataRDD.map(x=>**new** Person(x.getAs("id"),x.getAs("name"),x.getAs("age"))).collect()  
 **for**(p <- result) {  
 System.*out*.println(p.toString)  
 }  
  
 }  
}