Scala第十二章节

章节目标

1. 掌握Source读取数据的功能
2. 掌握写入数据的功能
3. 掌握学员成绩表案例

1. 读取数据

在Scala语言的`Source单例对象中`中, 提供了一些非常便捷的方法, 从而使开发者可以快速的从指定数据源(文本文件, URL地址等)中获取数据, 在使用`Source单例对象`之前, 需要先导包, 即`import scala.io.Source`.

1.1 按行读取

我们可以以`行`为单位, 来读取数据源中的数据, 返回值是一个`迭代器类型的对象`. 然后通过`toArray, toList`方法, 将这些数据放到数组或者列表中即可.

> 注意: Source类扩展自Iterator[Char]

格式

```
//1. 获取数据源文件对象.
val source:BufferedSource = Source.fromFile("数据源文件的路径","编码表")
//2. 以行为单位读取数据.
val lines:Iterator[String] = source.getLines()
//3. 将读取到的数据封装到列表中.
val list1:List[String] = lines.toList
//4. 千万别忘记关闭Source对象.
source.close()
```

需求

1. 在当前项目下创建data文件夹, 并在其中创建1.txt文本文件, 文件内容如下:

```
好好学习，天天向上!
Hadoop，Zookeeper，Flume，Spark
Flink，Sqoop，HBase
选择黑马，成就你一生的梦想.
```

2. 以行为单位读取该文本文件中的数据, 并打印结果.

参考代码

```
import scala.io.Source

//案例: 演示读取行.
```

```
object ClassDemo01 {
  def main(args: Array[String]): Unit = {
    //1. 获取数据源对象.
    val source = Source.fromFile("./data/1.txt")
    //2.通过getLines()方法, 逐行获取文件中的数据.
    var lines: Iterator[String] = source.getLines()
    //3. 将获取到的每一条数据都封装到列表中.
    val list1 = lines.toList
    //4. 打印结果
    for(s <- list1) println(s)
    //5. 记得关闭source对象.
    source.close()
  }
}
```

1.2 按字符读取

Scala还提供了`以字符为单位读取数据`这种方式, 这种用法类似于迭代器, 读取数据之后, 我们可以通过`hasNext()`, `next()方法`, 灵活的获取数据.

格式

```
//1. 获取数据源文件对象.
val source:BufferedSource = Source.fromFile("数据源文件的路径","编码表")
//2. 以字符为单位读取数据.
val iter:BufferedIterator[Char] = source.buffered
//3. 将读取到的数据封装到列表中.
while(iter.hasNext) {
  print(iter.next())
}
//4. 千万别忘记关闭Source对象.
source.close()
```

> 注意:
>
> 如果文件不是很大, 我们可以直接把它读取到一个字符串中.
>
> ```
> val str:String = source.mkString
> ```

需求

1. 在当前项目下创建data文件夹, 并在其中创建1.txt文本文件, 文件内容如下:

   ```
   好好学习，天天向上!
   Hadoop，Zookeeper，Flume，Spark
   Flink，Sqoop，HBase
   选择黑马，成就你一生的梦想.
   ```

2. 以行为单位读取该文本文件中的数据, 并打印结果.

参考代码

```
import scala.io.Source

//案例：演示读取单个字符.
object ClassDemo02 {
  def main(args: Array[String]): Unit = {
    //1. 获取数据源对象.
    val source = Source.fromFile("./data/1.txt")
    //2. 获取数据源文件中的每一个字符.
    val iter = source.buffered      //这里，source对象的用法相当于迭代器.
    //3. 通过hasNext()，next()方法获取数据.
    while(iter.hasNext) {
      print(iter.next())            //细节，这里不要用println()，否则输出结果可能不是你想要的.
    }

    //4. 通过mkString方法，直接把文件中的所有数据封装到一个字符串中.
    val str = source.mkString
    //5. 打印结果.
    println(str)
    //6. 关闭source对象，节约资源，提高效率.
    source.close()
  }
}
```

1.3 读取词法单元和数字

所谓的词法单元指的是`以特定符号间隔开的字符串`，如果数据源文件中的数据都是`数字形式的字符串`，我们可以很方便的从文件中直接获取这些数据，例如：

```
10 2 5
11 2
5 1 3 2
```

格式

```
//1. 获取数据源文件对象.
val source:BufferedSource = Source.fromFile("数据源文件的路径","编码表")
//2. 读取词法单元.
 // \s表示空白字符(空格，\t，\r，\n等)
val arr:Array[String] = source.mkString.split("\\s+")
//3. 将字符串转成对应的整数
val num = strNumber.map(_.toInt)
//4. 千万别忘记关闭Source对象.
source.close()
```

需求

1. 在当前项目下创建data文件夹，并在其中创建2.txt文本文件，文件内容如下：

```
10 2 5
11 2
5 1 3 2
```

2. 读取文件中的所有整数, 将其加1后, 把结果打印到控制台.

参考代码

```
import scala.io.Source

//案例: 读取词法单元和数字.
object ClassDemo03 {
  def main(args: Array[String]): Unit = {
    val source = Source.fromFile("./data/2.txt")
    // \s表示空白字符(空格, \t, \r, \n等)
    val strNumber = source.mkString.split("\\s+")

    //将字符串转成对应的整数
    val num = strNumber.map(_.toInt)
    for(a <- num) println(a + 1)
  }
}
```

1.4 从URL或者其他源读取数据

Scala中提供了一种方式, 可以让我们直接从指定的URL路径, 或者其他源(例如: 特定的字符串)中直接读取数据。

格式

- 从URL地址中读取数据

```
//1. 获取数据源文件对象.
val source = Source.fromURL("http://www.itcast.cn")
//2. 将数据封装到字符串中并打印.
println(source.mkString)
```

- 从其他源读取数据

```
//1. 获取数据源文件对象.
val str = Source.fromString("黑马程序员")
println(str.getLines())
```

需求

1. 读取`传智播客官网(http://www.itcast.cn)`页面的数据, 并打印结果.
2. 直接读取字符串`黑马程序员`, 并打印结果.

参考代码

```
import scala.io.Source

//案例: 从URL或者其他源读取数据
object ClassDemo04 {
  def main(args: Array[String]): Unit = {
    val source = Source.fromURL("http://www.itcast.cn")
    println(source.mkString)

    val str = Source.fromString("黑马程序员")
    println(str.getLines())
  }
}
```

1.5 读取二进制文件

Scala没有提供读取二进制文件的方法, 我们需要通过Java类库来实现.

需求

已知项目的data文件夹下有`05.png`这张图片, 请读取该图片数据, 并将读取到的字节数打印到控制台上.

参考代码

```
//案例: 读取二进制文件数据.
object ClassDemo05 {
  def main(args: Array[String]): Unit = {
    val file = new File("./data/04.png")
    val fis = new FileInputStream(file)
    val bys = new Array[Byte](file.length().toInt)
    fis.read(bys)
    fis.close()
    println(bys.length)
  }
}
```

2. 写入数据

Scala并没有内建的对写入文件的支持, 要写入数据到文件, 还是需要使用Java的类库.

2.1 往文件中写入指定数据

需求

往项目下的data文件夹的3.txt文本文件中, 编写一句话, 内容如下:

```
好好学习,
天天向上!
```

参考代码

```
//案例: 写入数据到文本文件.
object ClassDemo06 {
  def main(args: Array[String]): Unit = {
    val pw = new FileOutputStream("./data/3.txt")
    pw.write("好好学习,\r\n".getBytes())
    pw.write("天天向上!".getBytes())
    pw.close()
  }
}
```

2.2 序列化和反序列化

在Scala中, 如果想将对象传输到其他虚拟机, 或者临时存储, 就可以通过`序列化和反序列化`来实现了.

- 序列化: 把对象写到文件中的过程.
- 反序列化: 从文件中加载对象的过程.

> 注意: 一个类的对象要想实现序列化和反序列化操作, 则该类必须继承`Serializable特质`.

需求:

1. 定义样例类Person, 属性为姓名和年龄.
2. 创建Person样例类的对象p.
3. 通过序列化操作将对象p写入到项目下的data文件夹下的4.txt文本文件中.
4. 通过反序列化操作从项目下的data文件夹下的4.txt文件中, 读取对象p.

参考代码

```
//案例: 演示序列化和反序列化操作.
object ClassDemo07 {
  case class Person(var name:String, var age:Int)

  def main(args: Array[String]): Unit = {
    //序列化操作.
    /*val p = new Person("张三", 23)
    val oos = new ObjectOutputStream(new FileOutputStream("./data/4.txt"))
    oos.writeObject(p)
    oos.close()*/

    //反序列化操作.
    val ois = new ObjectInputStream(new FileInputStream("./data/4.txt"))
    var p: Person = ois.readObject().asInstanceOf[Person]
    println(p)

  }
}
```

3. 案例: 学员成绩表

7.1 概述

1. 已知项目下的data文件夹的student.txt文本文件中, 记录了一些学员的成绩, 如下:

   > 格式为: 姓名 语文成绩 数学成绩 英语成绩

   ```
   张三 37 90 100
   李四 90 73 81
   王五 60 90 76
   赵六 89 21 72
   田七 100 100 100
   ```

2. 按照学员的总成绩降序排列后, 按照`姓名 语文成绩 数学成绩 英语成绩 总成绩`的格式, 将数据写到项目下的data文件夹的stu.txt文件中.

7.2 目的

考察`流，样例类，以及函数式编程`相关内容.

7.3 步骤

1. 定义样例类Person, 属性为: 姓名, 语文成绩, 数学成绩, 英语成绩, 且该类中有一个获取总成绩的方法.
2. 读取指定文件(./data/student.txt)中所有的数据, 并将其封装到List列表中.
3. 定义可变的列表ListBuffer[Student], 用来记录所有学生的信息.
4. 遍历第二步获取到的数据, 将其封装成Person类的对象后, 并添加到ListBuffer中.
5. 对第4步获取到的数据进行排序操作, 并将其转换成List列表.
6. 按照指定格式, 通过BufferWriter将排序后的数据写入到目的地文件中(./data/stu.txt)
7. 关闭流对象.

7.4 参考代码

```
//案例: 按照学员的总分降序排列.
object ClassDemo08 {
  //1. 定义样例类Person, 属性: 姓名, 语文成绩, 数学成绩, 英语成绩, 且该类中有一个获取总成绩的方法.
  case class Student(name:String, chinese:Int, math:Int, english:Int) {
    def getSum() = chinese + math + english
  }

  def main(args: Array[String]): Unit = {
    //2. 获取数据源文件对象.
    val source = Source.fromFile("./data/student.txt")
    //3. 读取指定文件(./data/student.txt)中所有的数据, 并将其封装到List列表中.
    var studentList: Iterator[List[String]] = source.getLines().map(_.split("
")).map(_.toList)
    //4. 定义可变的列表ListBuffer[Student], 用来记录所有学生的信息.
    val list = new ListBuffer[Student]()
    //5. 遍历第二步获取到的数据, 将其封装成Person类的对象后, 并添加到ListBuffer中.
    for(s <- studentList) {
      list += Student(s(0), s(1).toInt, s(2).toInt, s(3).toInt)
    }
    //6. 对第5步获取到的数据进行排序操作, 并将其转换成List列表.
    val sortList = list.sortBy(_.getSum()).reverse.toList
```

```
    //7. 按照指定格式，通过BufferWriter将排序后的数据写入到目的地文件中(./data/stu.txt)
    val bw = new BufferedWriter(new FileWriter("./data/stu.txt"))
    for(s <- sortList) bw.write(s"${s.name} ${s.chinese} ${s.math} ${s.english}
${s.getSum()}\r\n")
    //8. 关闭流对象
    bw.close()
  }
}
```