Atitit 数据算法 聚合算法 stream jdk

目录

[1.1. 聚合为sum（） 1](#_Toc28256)

[1.2. Map映射转换数据类型int 1](#_Toc9376)

[1.3. 聚合为sum（） 2](#_Toc1572)

[2. 2.聚合操作和迭代器之间的区别 3](#_Toc11772)

## 聚合为sum（）

double all\_sale\_tax = ((List<Map>) input1.get("shoplist")).stream().mapToDouble(i -> (double) i.get("sale\_tax")).sum();  
DecimalFormat df2 = new DecimalFormat("###.00");//这样为保持2位  
input1.put("all\_sale\_tax", df2.format(all\_sale\_tax + 0.05));

## Map映射转换数据类型int

**private** **static** Object scoreSum(List<Map> rzt) {

**return** rzt.stream().mapToInt(e -> {

System.***out***.println(e);

**return** (Integer) e.get("score");

// e.get("score")==null?0:(Integer) e.get("score");

}).sum();

}

## 聚合为sum（）

　以下示例使用****一个包含聚合操作filter、mapToInt和average的管道****计算集合roster中所包含的所有男性成员的平均年龄：

double average = roster

.stream() *// Stream< Person >*

.filter(p -> p.getGender() == Person.Sex.MALE) *// Stream< Person >*

.mapToInt(Person::getAge) *// IntStream*

.average() *// OptionalDouble*

.getAsDouble(); *// Double*

# 2.聚合操作和迭代器之间的区别

聚合操作，如forEach，就像是迭代器一样。但是，它们有几个基本的区别：

* ****聚合操作使用内部迭代****：聚合操作不包含像next这样指示它们处理集合的下一个元素的方法。当使用内部委派时，您的应用程序决定哪些集合进行迭代，但是由JDK决定如何遍历集合。当使用外部迭代时，您的应用程序将决定迭代的集合以及如何迭代它。但是，外部迭代只能依次迭代集合的元素。而内部迭代没有这个限制。它可以更容易地利用****并行计算****，这涉及到将问题分解为子问题，同时解决这些问题，然后将子问题的结果结合起来。更多有关信息请参见[官方文档](https://docs.oracle.com/javase/tutorial/collections/streams/parallelism.html" \o "Parallelism" \t "https://blog.csdn.net/why_still_confused/article/details/_blank) 。
* ****聚合操作从流中处理元素****：聚合操作从流中处理元素，而不是直接从集合中处理。因此，他们也被称为****流操作****。
* ****聚合操作支持将行为作为参数****：在大多数聚合操作中，可以指定lambda表达式为参数。这使您可以自定义特定聚合操作的行为。