Java 核心技术(高阶)

第八章 Java Stream

第五节 流的计算结果

华东师范大学 陈良育

Stream的工作流程

- 流的工作流程
  - 创建一个流
  - 指定将流转换为其他流的中间操作，可包括多个步骤(惰性操作)
  - 应用终止操作，从而产生结果。这个操作会强制执行之前的惰性操作。这个步骤以后，流就再也不用了。

例子

```
Integer[] a = new Integer[] {2,4,6,8};

Stream<Integer> s1 = Stream.of(a);
long countResult = s1.count();
System.out.println("the count result of s1 is " + countResult);

Stream<Integer> s2 = Stream.of(a);
countResult = s2.filter(n-> n>10).count();
System.out.println("the count result of s2 is " + countResult);
```

流的计算结果(1)

- 流的计算
  - 简单约简(聚合函数)：count/max/min/…
  - 自定义约简：reduce
  - 查看/遍历元素：iterator/forEach
  - 存放到数据结构中

流的计算结果(2)

- 流的计算：简单约简(聚合函数)
  - count(), 计数
  - max(Comparator)，最大值，需要比较器
  - min(Comparator)，最小值，需要比较器
  - findFirst(), 找到第一个元素
  - findAny(), 找到任意一个元素
  - anyMatch(Predicate)，如有任意一个元素满足Predicate，返回true
  - allMatch(Predicate)，如所有元素满足Predicate，返回true
  - noneMatch(Predicate)，如没有任何元素满足Predicate，返回true

流的计算结果(3)

- 流的计算：自定义约简
  - reduce，传递一个二元函数BinaryOperator，对流元素进行计算
  - 如求和、求积、字符串连接等

```
Integer[] a = new Integer[] {2,4,6,8};

Stream<Integer> s1 = Stream.of(a);
Optional<Integer> sum = s1.reduce(Integer::sum);
System.out.println(sum.get());

Stream<Integer> s2 = Stream.of(a);
Optional<Integer> product = s2.reduce((x,y)->x*y);
System.out.println(product.get());

Stream<Integer> s3 = Stream.of(a);
Integer product3 = s3.reduce(1,(x,y)->x*y);
System.out.println(product3);
```

流的计算结果(4)

- 流的计算：查看/遍历元素
  - iterator()，遍历元素
  - forEach(Consumer)，应用一个函数到每个元素上

```
Integer[] a = new Integer[] {2,4,6,8};

Stream<Integer> s1 = Stream.of(a);
Iterator<Integer> it1 = s1.filter(n->n>2).iterator();
while(it1.hasNext()) {
    System.out.println(it1.next());
}

Stream<Integer> s2 = Stream.of(a);
s2.filter(n->n>2).forEach(System.out::println);
```

流的计算结果(5)

- 流的计算：存放到数据结构中
  - toArray()，将结果转为数组
  - collect(Collectors.toList()),将结果转为List
  - collect(Collectors.toSet()),将结果转为Set
  - collect(Collectors.toMap()), 将结果转为Map
  - collect(Collectors.joining()), 将结果连接起来

流的计算结果(6)

- 流的高阶计算:
  - 分组groupingBy和分区partitionBy
  - 分组后的约简
    - counting
    - summing
    - maxBy
    - minBy
  - 以上方法均在java.util.stream.Collectors中
  - 请查看《Java核心技术》下卷高级特性第一章

总结

- 流的计算
  - 流的最终运算结果
  - 惰性计算，启动先前的转换环节
  - 三种结果
    - 约简
    - 遍历/查看
    - 转存为其他的数据结构

谢 谢！