Java 核心技术(高阶)

第八章 Java Stream

第六节 流的应用

华东师范大学 陈良育

Stream的工作流程

- Stream的工作流程
  - 创建一个流
  - 将流转换为其他流的中间操作，可包括多个步骤(惰性操作，lazy)
  - 应用终止操作，从而产生结果
  - 只有第三步启动，才会执行第二步的惰性操作

Stream的优点

- Java Stream的优点
  - 统一转换元素
  - 过滤元素
  - 利用单个操作合并元素
  - 将元素序列存放到某一个集合中
  - 搜索满足某些条件的元素的序列
  - 类似SQL操作，遵循“做什么而非怎么做”原则
  - 简化了串行/并行的大批量操作

Stream和循环迭代代码的区别

- Java Stream vs 循环迭代代码
  - Stream广泛使用Lambda表达式，只能读取外围的final或者effectively final变量，循环迭代代码可以读取/修改任意的局部变量。
  - 在循环迭代代码块中，可以随意break/continue/return，或者抛出异常，而Lambda表达式无法完成这些事情。
  - Stream流不是淘汰循环迭代代码，应该是两者相互搭配使用。

Stream应用注意事项(1)

- 一个流，一次只能一个用途，不能多个用途，用了不能再用

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

Stream<Integer> s1 = Stream.of(a);
Stream<Integer> s2 = s1.filter(x->x>4);
//Stream<Integer> s3 = s1.filter(x->x>2);   //error

//s1.forEach(System.out::println);                //error
s2.forEach(System.out::println);
//s2.forEach(System.out::println);                  //error
```

java.lang.IllegalStateException: stream has already been operated upon or closed
tractPipeline.sourceStageSpliterator(AbstractPipeline.java:274)

Stream应用注意事项(2)

- 避免创建无限流

```
//无限流
IntStream.iterate(0, i -> i + 1)
        .forEach(System.out::println);

//需要对流元素进行限制
IntStream.iterate(0, i -> i + 1)
.limit(10).forEach(System.out::println);

//又一个无限流
IntStream.iterate(0, i -> ( i + 1 ) % 2)
.distinct().limit(10).forEach(System.out::println);
```

Stream应用注意事项(3)

- 注意操作顺序

```
IntStream.iterate(0, i -> i + 1)
.limit(10) // LIMIT
.skip(5) // OFFSET
.forEach(System.out::println); //5-9

IntStream.iterate(0, i -> i + 1)
.skip(5) // OFFSET
.limit(10) // LIMIT
.forEach(System.out::println); //5-14
```

Stream应用注意事项(4)

- 谨慎使用并行流
  - 底层使用Fork-Join Pool，处理计算密集型任务
  - 数据量过小不用
  - 数据结构不容易分解的时候不用，如LinkedList等
  - 数据频繁拆箱装箱不用
  - 涉及findFirst或者limit的时候不用

```
IntStream s1 = IntStream.range(1,10000000);
long evenNum = s1.parallel().filter(n->n%2==0).count();
System.out.println(evenNum);
```

Stream应用注意事项(5)

- Stream vs Collection
  - Stream和Collection两者可以互相转化
  - 如果数据可能无限，用Stream
  - 如果数据很大很大，用Stream
  - 如果调用者将使用查找/过滤/聚合等操作，用Stream
  - 当调用者使用过程中，发生数据改变，而调用者需要对数据一致性有较高要求，用Collection
  - https://stackoverflow.com/questions/24676877/should-i-return-a-collection-or-a-stream/24679745#24679745

总结

- Java Stream
  - Stream很强大，效率很高
  - 需要认真学习，小心使用

谢 谢！