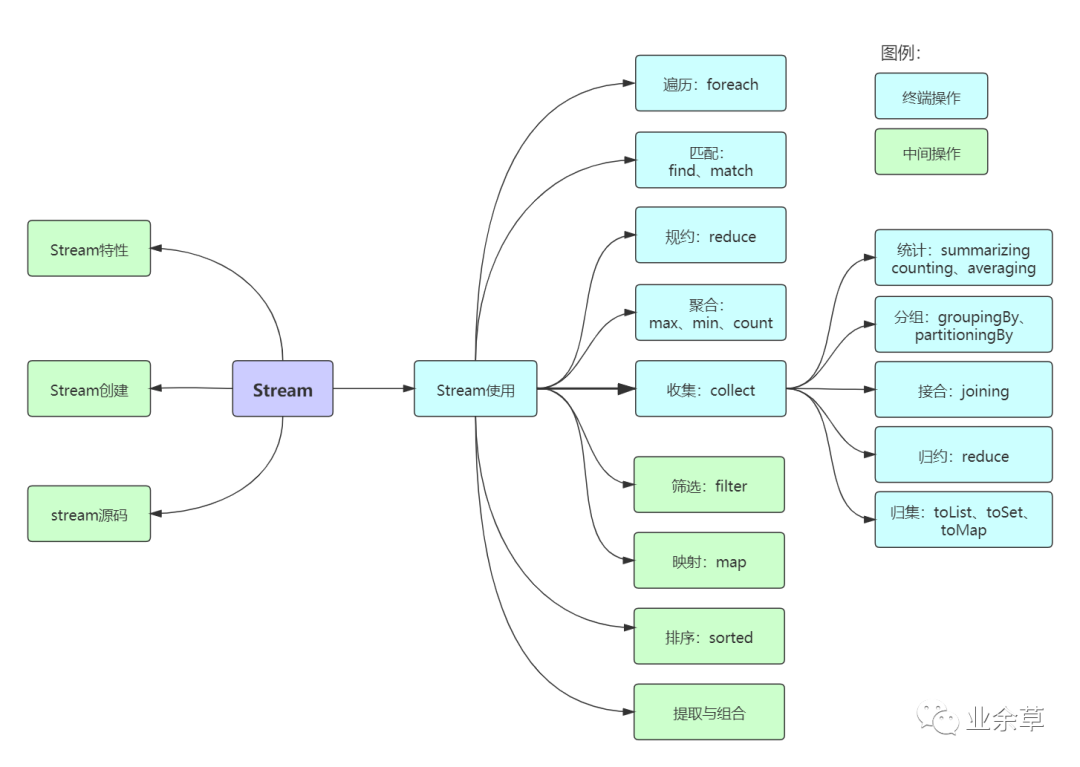
**Java 8 Stream：20 个实例，玩转集合的筛选、归约、分组、聚合**

<https://mp.weixin.qq.com/s/cnPi3d9XTGcqV6oqRgBJ8g>



先贴上几个案例，水平高超的同学可以挑战一下：

1. 从员工集合中筛选出salary大于8000的员工，并放置到新的集合里。
2. 统计员工的最高薪资、平均薪资、薪资之和。
3. 将员工按薪资从高到低排序，同样薪资者年龄小者在前。
4. 将员工按性别分类，将员工按性别和地区分类，将员工按薪资是否高于8000分为两部分。

用传统的迭代处理也不是很难，但代码就显得冗余了，跟Stream相比高下立判。

**1 Stream概述**

Java 8 是一个非常成功的版本，这个版本新增的Stream，配合同版本出现的 Lambda ，给我们操作集合（Collection）提供了极大的便利。

那么什么是Stream？

**❝**

Stream将要处理的元素集合看作一种流，在流的过程中，借助Stream API对流中的元素进行操作，比如：筛选、排序、聚合等。

❞

Stream可以由数组或集合创建，对流的操作分为两种：

1. 中间操作，每次返回一个新的流，可以有多个。
2. 终端操作，每个流只能进行一次终端操作，终端操作结束后流无法再次使用。终端操作会产生一个新的集合或值。

另外，Stream有几个特性：

1. stream不存储数据，而是按照特定的规则对数据进行计算，一般会输出结果。
2. stream不会改变数据源，通常情况下会产生一个新的集合或一个值。
3. stream具有延迟执行特性，只有调用终端操作时，中间操作才会执行。

**2 Stream的创建**

Stream可以通过集合数组创建。

1、通过 java.util.Collection.stream() 方法用集合创建流

2、使用java.util.Arrays.stream(T[] array)方法用数组创建流

3、使用Stream的静态方法：of()、iterate()、generate()

如果流中的数据量足够大，并行流可以加快处速度。

除了直接创建并行流，还可以通过parallel()把顺序流转换成并行流：

## 3 Stream的使用

在使用stream之前，先理解一个概念：Optional 。

**❝**

Optional类是一个可以为null的容器对象。如果值存在则isPresent()方法会返回true，调用get()方法会返回该对象。

**3.1 遍历/匹配（foreach/find/match）**

**3.2 筛选（filter）**

**3.3 聚合（max/min/count)**

**3.4 映射(map/flatMap)**

**3.5 归约(reduce)**

**3.6 收集(collect)**

### 3.6.1 归集(toList/toSet/toMap)

### 3.6.2 统计(count/averaging)

Collectors提供了一系列用于数据统计的静态方法：

* 计数：count
* 平均值：averagingInt、averagingLong、averagingDouble
* 最值：maxBy、minBy
* 求和：summingInt、summingLong、summingDouble
* 统计以上所有：summarizingInt、summarizingLong、summarizingDouble

### 3.6.3 分组(partitioningBy/groupingBy)

### 3.6.4 接合(joining)

### 3.6.5 归约(reducing)

**3.7 排序(sorted)**

**3.8 提取/组合**