JDK8新特性

Stream流式编程

Java8中有两大最为重要的改变。第一个是 Lambda 表达式；另外一个则是 Stream API。

Stream API (java.util.stream) 把真正的函数式编程风格引入到Java中。这是目前为止对Java类库最好的补充，因为Stream API可以极大提高Java程序员的生产力，让程序员写出高效率、干净、简洁的代码。

Stream 是 Java8 中处理集合的关键抽象概念，它可以指定你希望对集合进行的操作，可以执行非常复杂的查找、过滤和映射数据等操作。 使用Stream API 对集合数据进行操作，就类似于使用 SQL 执行的数据库查询。也可以使用 Stream API 来并行执行操作。简言之，Stream API 提供了一种高效且易于使用的处理数据的方式。

Stream是数据渠道，用于操作数据源（集合、数组等）所生成的元素序列。“集合讲的是数据，负责存储数据，Stream流讲的是计算，负责处理数据！”

注意：

①Stream 自己不会存储元素。

②Stream 不会改变源对象。每次处理都会返回一个持有结果的新Stream。

③Stream 操作是延迟执行的。这意味着他们会等到需要结果的时候才执行。

1- 创建 Stream：通过一个数据源（如：集合、数组），获取一个流

2- 中间操作：每次处理都会返回一个持有结果的新Stream，即中间操作的方法返回值仍然是Stream类型的对象，因此中间操作可以是个操作链，可对数据源的数据进行n次处理，但是在终结操作前，并不会真正执行。

3- 终止操作：终止操作的方法返回值类型就不再是Stream了，因此一旦执行终止操作，就结束整个Stream操作了。一旦执行终止操作，就执行中间操作链，最终产生结果并结束Stream。

创建方式

`创建 Stream方式一：通过集合`

Java8 中的 Collection 接口被扩展，提供了两个获取流的方法： default Stream stream() : 返回一个顺序流

`创建 Stream方式二：通过数组`

Java8 中的 Arrays 的静态方法 stream() 可以获取数组流：

static Stream stream(T[] array): 返回一个流

`创建 Stream方式三：通过Stream的of()`

可以调用Stream类静态方法 of(), 通过显示值创建一个流。它可以接收任意数量的参数。

public static Stream of(T... values) : 返回一个流

`创建 Stream方式四：创建无限流（了解）`

可以使用静态方法 Stream.iterate() 和 Stream.generate(), 创建无限流。

public static Stream generate(Supplier s)

中间操作

方法	描述
`filter(Predicate p)`	保存符合指定条件的元素
`distinct()`	筛选，通过流所生成元素的equals() 去除重复元素
`limit(long maxSize)`	保留指定个数的前
`skip(long n)`	跳过元素，返回一个扔掉了前 n 个元素的流。若流中元素不足 n 个，则返回一个空流。与 limit(n) 互补
`sorted()`	产生一个新流，其中按自然顺序排序
`map(Function f)`	接收一个函数作为参数，该函数会被应用到每个元素上，并将其映射成一个新的元素。
`flatMap(Function f)`	接收一个函数作为参数，将流中的每个值都换成另一个流，然后把所有流连接成一个流

终止操作

方法	描述
boolean allMatch(Predicate p)	检查是否匹配所有元素
boolean anyMatch(Predicate p)	检查是否至少匹配一个元素
boolean noneMatch(Predicate p)	检查是否没有匹配所有元素
Optional findFirst()	返回第一个元素
long count()	返回流中元素总数
Optional max()	返回流中最大值
Optional min()	返回流中最小值
void forEach(Consumer c)	迭代

7. Optional类

> 到目前为止，臭名昭著的空指针异常是导致Java应用程序失败的最常见原因。以前，为了解决空指针异常，Google公司著名的Guava项目引入了Optional类，Guava通过使用检查空值的方式来防止代码污染，它鼓励程序员写更干净的代码。受到Google Guava的启发，Optional类已经成为Java 8类库的一部分。
>
> Optional实际上是个容器：它可以保存类型T的值，或者仅仅保存null。Optional提供很多有用的方法，这样我们就不用显式进行空值检测。

方法	描述
of(Object obj)	根据传入对象获取一个Optional对象，此对象不能为null
empty()	包装一个保存有null的Optional对象
ofNullable(Object obj)	根据传入对象获取一个Optional对象，此对象可以为null
get()	获取Optional中保存的对象，如果为null，则报空指针异常
isPresent()	表示判断Optional是否为null，为null结果为false，不为null结果为true
ifPresent(Consumer<? super T> consumer)	如果Optional对象中的对象不为null，则消费此对象，否则不消费
orElse(T other)	如果当前Optional对象中保存对象为null，则使用传入对象
orElseGet(Supplier<? extends T> other)	如果当前Optional对象中为null则获取到另外一个对象，否则不获取
orElseThrow(Supplier<? extends X> exceptionSupplier)	如果当前Optional对象中为null则抛出异常，否则不抛出