# Java 9 改进的 Stream API

Java 9 改进的 Stream API 添加了一些便利的方法，使流处理更容易，并使用收集器编写复杂的查询。

Java 9 为 Stream 新增了几个方法：dropWhile、takeWhile、ofNullable，为 iterate 方法新增了一个重载方法。

### **takeWhile 方法**

**语法**

default Stream<T> takeWhile(Predicate<? super T> predicate)

takeWhile() 方法使用一个断言作为参数，返回给定 Stream 的子集直到断言语句第一次返回 false。如果第一个值不满足断言条件，将返回一个空的 Stream。

takeWhile() 方法在有序的 Stream 中，takeWhile 返回从开头开始的尽量多的元素；在无序的 Stream 中，takeWhile 返回从开头开始的符合 Predicate 要求的元素的子集。

## 实例

import java.util.stream.Stream; public class Tester { public static void main(String[] args) { Stream.of("a","b","c","","e","f").takeWhile(s->!s.isEmpty()) .forEach(System.out::print); } }

以上实例 takeWhile 方法在碰到空字符串时停止循环输出，执行输出结果为：

abc

### **dropWhile 方法**

**语法**

default Stream<T> dropWhile(Predicate<? super T> predicate)

dropWhile 方法和 takeWhile 作用相反的，使用一个断言作为参数，直到断言语句第一次返回 true 才返回给定 Stream 的子集。

## 实例

import java.util.stream.Stream; public class Tester { public static void main(String[] args) { Stream.of("a","b","c","","e","f").dropWhile(s-> !s.isEmpty()) .forEach(System.out::print); } }

以上实例 dropWhile 方法在碰到空字符串时开始循环输出，执行输出结果为：

ef

### **iterate 方法**

**语法**

static <T> Stream<T> iterate(T seed, Predicate<? super T> hasNext, UnaryOperator<T> next)

方法允许使用初始种子值创建顺序（可能是无限）流，并迭代应用指定的下一个方法。 当指定的 hasNext 的 predicate 返回 false 时，迭代停止。

## 实例

java.util.stream.IntStream; public class Tester { public static void main(String[] args) { IntStream.iterate(3, x -> x < 10, x -> x+ 3).forEach(System.out::println); } }

执行输出结果为：

369

### **ofNullable 方法**

**语法**

static <T> Stream<T> ofNullable(T t)

ofNullable 方法可以预防 NullPointerExceptions 异常， 可以通过检查流来避免 null 值。

如果指定元素为非 null，则获取一个元素并生成单个元素流，元素为 null 则返回一个空流。

## 实例

import java.util.stream.Stream; public class Tester { public static void main(String[] args) { long count = Stream.ofNullable(100).count(); System.out.println(count); count = Stream.ofNullable(null).count(); System.out.println(count); } }

执行输出结果为：

10