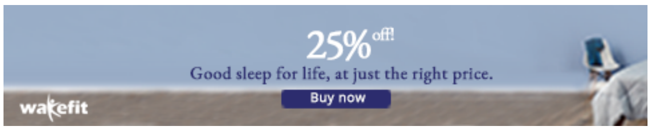


Java 8 Streams - Stateful vs Stateless behavioral parameters

[Updated: Feb 7, 2017, Created: Nov 6, 2016]

A stateful operation is the one whose result depends on any state that might change during the execution of the pipeline. Stateless operations retain no state during the execution of the pipeline.

Stateful intermediate operations

Java 8 stream API provides various intermediate operations which are stateful by definition. They maintain some state internally to accomplish the operation.

Those intermediate operations are **distinct()**, **sorted()**, **limit()**, **skip()**. All other operations are stateless.

These stateful intermediate operations incorporate state from previously processed elements when processing the current element.

Stateful behavioral parameter

A developer can choose to access/update an external state from a behavioral parameter.

A behavioral parameter is nothing but a functional interface or lambda expression used in intermediate operations.

The problem with stateful behavioral parameter is that: it may render the result nondeterministic, specially for parallel pipeline.

A nondeterministic operation produces different results for the same given input if executed multiple times.

Also accessing external state is not thread-safe and needs to be synchronized.

From Stream API specs :

> Note also that attempting to access mutable state from behavioral parameters presents you with a bad choice with respect to safety and performance; if you do not synchronize access to that state, you have a data race and therefore your code is broken, but if you do synchronize access to that state, you risk having contention undermine the parallelism you are seeking to benefit from. The best approach is to avoid stateful behavioral parameters to stream operations entirely; there is usually a way to restructure the stream pipeline to avoid statefulness.

KHAZANA JEWELLERY
DIWALI MEGA OFFER WITH SURPRISE GIFTS FROM KHAZANA
Gold, Diamond & Platinum jewellery
FREE HOME APPLIANCE
WITH EVERY PURCHASE
OFFER VALIDITY
28TH SEP - 27TH OCT 2019
BANGALORE: Dickenson Road, Jayanagar, Marathahalli, Malleshwaram

Examples

Following example shows the usage of stateful behavioral parameter. The pipeline finds the sum of the integers for distinct elements.

The whole process is repeated in a for loop multiple times to see how the results can be non-deterministic.

```java
public class StatefulExample {

    public static void main (String[] args) {
        for (int i = 0; i < 5; i++) {

            Set<Integer> seen = new HashSet<>();
            IntStream stream = IntStream.of(1, 2, 1, 2, 3, 4, 4, 5);
            int sum = stream.parallel().map(
                //stateful behavioral parameter.
                e -> {
                      if (seen.add(e))
                          return e;
                      else
                          return 0;
                  }).sum();

            System.out.println(sum);

        }
    }
}
```

Output

```
19
17
15
17
15
```

Above example can be fixed by using synchronized set, e.g. the one created with `Collections.synchronizedSet(new HashSet())` but that would undermine the benefit of parallelism.

We can always find a way to avoid stateful behavioral parameter. Above example can be fixed by replacing stateful lambda expression with distinct operation (which is internally stateful but is safe and deterministic)

```java
public class StatefulFixExample {

    public static void main (String[] args) {
        for (int i = 0; i < 5; i++) {
            IntStream stream = IntStream.of(1, 2, 1, 2, 3, 4, 4, 5);
            int sum = stream.parallel().distinct().sum();
            System.out.println(sum);
        }
    }
}
```

Output

```
15
15
15
15
15
```

Here's another example which finds the sum of even numbers and at the same time attempts to find the count of those even numbers by using an external count variable.

```java
public class StatefulExample2 {
    private static int count = 0;
```

```
    public static void main (String[] args) {
        for (int i = 0; i < 5; i++) {
            process();
        }
    }

     private static void process () {
        count = 0;

        IntStream stream = IntStream.range(1, 1000);
        //finding the sum of even numbers
        int sum = stream.parallel()
                        .filter(i -> {
                            boolean b = i % 2 == 0;
                            if (b) {
                                count++;//updating count hence making lambda stateful.
                            }
                            return b;
                        })
                        .sum();

        System.out.printf("sum :%d  count:%d%n", sum, count);
    }
}
```

Output

```
sum :249500  count:365
sum :249500  count:433
sum :249500  count:413
sum :249500  count:437
sum :249500  count:466
```

As seen in the output 'count' is not deterministic.

To fix above code, we are going to find the sum and count separately:

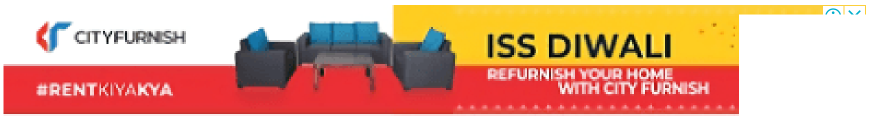
```
public class StatefulFixExample2 {
    public static void main (String[] args) {
        for (int i = 0; i < 5; i++) {
            process();
        }
    }

    private static void process () {
        IntStream stream = IntStream.range(1, 1000);

        //finding the even numbers
        int[] even = stream.parallel()
                           .filter(i -> i % 2 == 0)
                           .toArray();

        //finding sum
        int sum = IntStream.of(even).parallel().sum();

        System.out.printf("sum :%d  count:%d%n", sum, even.length);
    }
}
```

Output

L

Java 1

```
sum :249500  count:499
sum :249500  count:499
sum :249500  count:499
sum :249500  count:499
sum :249500  count:499
```



Dependencies and Technologies Used:

- JDK 1.8
- Maven 3.0.4

Stateful Vs Stateless Example

```
stateful-behavioral-parameter
    src
        main
            java
                com
                    logicbig
                        example
                            StatefulExample.java
                            StatefulExample2.java
                            StatefulFixExample.java
                            StatefulFixExample2.java
    pom.xml
```

```java
package com.logicbig.example;

import java.util.HashSet;
import java.util.Set;
import java.util.stream.IntStream;

public class StatefulExample {

    public static void main (String[] args) {
        for (int i = 0; i < 10; i++) {
            Set<Integer> seen = new HashSet<>();
            IntStream stream = IntStream.of(1, 2, 1, 2, 3, 4, 4, 5);
            int sum = stream.parallel().map(
                                //stateful behavioral parameter.
                                e -> {
                                    try {//making it bit slow for more thread
                                        //interference changes
                                        Thread.sleep(10);
                                    } catch (InterruptedException e1) {
                                        e1.printStackTrace();
                                    }
                                    if (seen.add(e))
                                        return e;
                                    else
                                        return 0;
```

Project Structure

```
stateful-behavioral-parameter
    src
        main
            java
                com
                    logicbig
                        example
                            StatefulExample.java
                            StatefulExample2.java
                            StatefulFixExample.java
                            StatefulFixExample2.java
    pom.xml
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.logicbig.example</groupId>
    <artifactId>stateful-behavioral-parameter</artifactId>
    <version>1.0-SNAPSHOT</version>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

```java
package com.logicbig.example;

import java.util.HashSet;
import java.util.Set;
import java.util.stream.IntStream;

public class StatefulExample {

    public static void main (String[] args) {
        for (int i = 0; i < 10; i++) {
            Set<Integer> seen = new HashSet<>();
            IntStream stream = IntStream.of(1, 2, 1, 2, 3, 4, 4, 5);
            int sum = stream.parallel().map(
                                //stateful behavioral parameter.
                                e -> {
                                    try {//making it bit slow for more thread
                                        //interference changes
                                        Thread.sleep(10);
                                    } catch (InterruptedException e1) {
                                        e1.printStackTrace();
                                    }
                                    if (seen.add(e))
                                        return e;
                                    else
                                        return 0;
                                }).sum();
            System.out.println(sum);

        }
    }
}
```

```java
package com.logicbig.example;


import java.util.stream.IntStream;

public class StatefulExample2 {
    private static int count = 0;
    public static void main (String[] args) {
        for (int i = 0; i < 5; i++) {
            process();
        }
    }

    private static void process () {
        count = 0;

        IntStream stream = IntStream.range(1, 1000);
        //finding the sum of even numbers
        int sum = stream.parallel()
                        .filter(i -> {
                            boolean b = i % 2 == 0;
                            if (b) {
                                count++;//updating count hence making it stateful.
                            }
                            return b;
                        })
                        .sum();

        System.out.printf("sum :%d  count:%d%n", sum, count);
    }
}
```

```
package com.logicbig.example;

import java.util.stream.IntStream;

public class StatefulFixExample {

    public static void main (String[] args) {
        for (int i = 0; i < 10; i++) {
            IntStream stream = IntStream.of(1, 2, 1, 2, 3, 4, 4, 5);
            int sum = stream.parallel().distinct().sum();
            System.out.println(sum);
        }
    }
}
```

```
package com.logicbig.example;

import java.util.stream.IntStream;


public class StatefulFixExample2 {
    public static void main (String[] args) {
        for (int i = 0; i < 5; i++) {
            process();
        }
    }

    private static void process () {
        IntStream stream = IntStream.range(1, 1000);

        //finding the even numbers
        int[] even = stream.parallel()
                           .filter(i -> i % 2 == 0)
                           .toArray();

        //finding sum
        int sum = IntStream.of(even).parallel().sum();

        System.out.printf("sum :%d  count:%d%n", sum, even.length);
    }
}
```

See Also

Ordering

Short circuiting operations

Lazy evaluation

Sequential vs Parallel streams

Java 8 Stream operations cheat sheet

What are Java 8 streams?

Side Effects

Reduction

Mutable Reduction

Java 8 Streams quick examples

Java 11 Tutorials

Java 11 Features

Java 10 Tutorials

Java 10 Features

Java 9 Tutorials

Java 9 Module System

Java 9 Misc Features

Java 9 JShell

Recent Tutorials

Spring Cloud - Hystrix CircuitBreaker, Thread Local Context Propagation

Spring Cloud - Circuit Breaker Hystrix Event Listener

Spring Cloud - Circuit Breaker Hystrix, Changing Default Thread Pool Properties

Spring Cloud - Circuit Breaker Hystrix, concurrent requests and default thread pool size

Spring Cloud - Circuit Breaker, Specifying Hystrix configuration in application.properties file

Spring Cloud - Hystrix Circuit Breaker, Setting Configuration Properties Using @HystrixProperty

Spring Cloud - Hystrix Circuit Breaker, getting failure exception in fallback method

Spring Cloud - Circuit Breaker Hystrix Basics

TypeScript - Standard JavaScript's built-in objects Supp

JavaScript - Async Processing with JavaScript Promise

TypeScript - Applying Mixins

JPA Criteria API - Modifying and Reusing Query Objects

JPA Criteria API - Delete Operations

JPA Criteria API - Update Operations

JPA Criteria API - Case Expressions with CriteriaBuilder.s

Share