Java 8 Stream and Lambda Expressions – Parsing File Example

(/users/183661/eyalgo.html) by Eyal Golan (/users/183661/eyalgo.html) MVB · Jan. 12, 15 · Java Zone (/java-jdk-development-tutorials-tools-news)

Recently I wanted to extract certain data from an output log.
Here's part of the log file:

```
2015-01-06 11:33:03 b.s.d.task [INFO] Emitting: eVentToRequestsBolt __ack_ack [-6722594615019711369 -1335723027906100557]
2015-01-06 11:33:03 c.s.p.d.PackagesProvider [INFO] ===---> Loaded package com.foo.bar
2015-01-06 11:33:04 b.s.d.executor [INFO] Processing received message source: eventToManageBolt:2, stream: __ack_ack, id: {}, [-67225946150197
11369 -1335723027906100557]
2015-01-06 11:33:04 c.s.p.d.PackagesProvider [INFO] ===---> Loaded package co.il.boo
2015-01-06 11:33:04 c.s.p.d.PackagesProvider [INFO] ===---> Loaded package dot.org.biz
```

I decided to do it using the Java8 Stream and Lambda Expression features.

Read the file
First, I needed to read the log file and put the lines in a Stream:

```
Stream<String> lines = Files.lines(Paths.get(args[1]));
```

Filter relevant lines
I needed to get the packages names and write them into another file.

Not all lines contained the data I need, hence *filter* only relevant ones.

```
lines.filter(line -> line.contains("===---> Loaded package"))
```

Parsing the relevant lines
Then, I needed to parse the relevant lines.
I did it by first splitting each line to an array of Strings and then taking the last element in that array.
In other words, I did a double *mapping*. First a line to an array and then an array to a String.

```
.map(line -> line.split(" "))
.map(arr -> arr[arr.length - 1])
```

Writing to output file
The last part was taking each string and write it to a file. That was the *terminal* operation.

```
.forEach(package -> writeToFile(fw, package));
```

writeToFile is a method I created.
The reason is that Java File System throws IOException. You can't use checked exceptions in lambda expressions.

Here's a full example (note, I don't check input)

```
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Stream;

public class App {
    public static void main(String[] args) throws IOException {
        Stream<String> lines = null;
        if (args.length == 2) {
            lines = Files.lines(Paths.get(args[1]));
        } else {
            String s1 = "2015-01-06 11:33:03 b.s.d.task [INFO] Emitting: adEventToRequestsBolt __ack_ack [-67225946150197113369 -13357230279061
00557]";
            String s2 = "2015-01-06 11:33:03 b.s.d.executor [INFO] Processing received message source: eventToManageBolt:2, stream: __ack_ack,
 id: {}, [-6722594615019711369 -1335723027906100557]";
            String s3 = "2015-01-06 11:33:04 c.s.p.d.PackagesProvider [INFO] ===---> Loaded package com.foo.bar";
            String s4 = "2015-01-06 11:33:04 c.s.p.d.PackagesProvider [INFO] ===---> Loaded package co.il.boo";
            String s5 = "2015-01-06 11:33:04 c.s.p.d.PackagesProvider [INFO] ===---> Loaded package dot.org.biz";
            List<String> rows = Arrays.asList(s1, s2, s3, s4, s5);
            lines = rows.stream();
        }

        new App().parse(lines, args[0]);

    }

    private void parse(Stream<String> lines, String output) throws IOException {
        final FileWriter fw = new FileWriter(output);

        //@formatter:off
        lines.filter(line -> line.contains("===---> Loaded package"))
        .map(line -> line.split(" "))
        .map(arr -> arr[arr.length - 1])
        .forEach(package -> writeToFile(fw, package));
        //@formatter:on
        fw.close();
        lines.close();
    }

    private void writeToFile(FileWriter fw, String package) {
        try {
            fw.write(String.format("%s%n", package));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

}
```


Topics:

Published at DZone with permission of Eyal Golan , DZone MVB . (http://eyalgo.com/2015/01/06/java-8-stream-and-lambda-expressions-parsing-file-example/)