HowToDoInJava

Java Stream forEach()

Last Updated: March 15, 2022 By: Lokesh Gupta Java 8 Java Stream Basics, Java Stream Methods

Java Stream *forEach()* method is used to **iterate over all the elements of the given Stream and to perform an `Consumer` *action* on each element of the Stream**.

The *forEach()* is a more concise way to write the for-each loop statements.

1. Stream forEach() Method

1.1. Method Syntax

The `forEach()` method syntax is as follows:

Syntax

```
void forEach(Consumer<? super T> action)
```

`Consumer` is a functional interface and `action` represents a non-interfering action to be performed on each element in the Stream. It accepts an input and returns no result.

1.2. Description

- The `forEach()` method is a **terminal operation**. It means that it does not return an output of type `Stream`.

- After **forEach()** is performed, the stream pipeline is considered consumed, and Stream can no longer be used.
- If we need to traverse the same data source again (the collection backing the Stream), we must return to the data source to get a new stream.
- For *parallel streams*, the `forEach()` operation does not guarantee the order of elements in the stream, as doing so would sacrifice the benefit of parallelism.
- If the provided Consumer `action` accesses the shared state between the Stream elements the `action` is responsible for providing the required synchronization.

2. Stream forEach() Examples

Example 1: Traversing the elements of a Stream and printing them

In this Java example, we are iterating over a `Stream` of Integers and printing all the integers to the standard output.

Stream forEach() Example

```
List<Integer> list = Arrays.asList(2, 4, 6, 8, 10);
Consumer<Integer> action = System.out::println;

list.stream()
    .forEach( action );
```

Note that we can write the above iteration using the *enhanced for-loop* as well.

Same iteration Using enhanced for loop

```
for (Integer i : list) {
  System.out.println(i);
}
```

Example 2: Traversing the elements in reverse order and printing them

Java example to iterate over stream elements and print them in reverse order.

Stream forEach() in Reverse Order

```
List<Integer> list = Arrays.asList(2, 4, 6, 8, 10);

list.stream()
    .sorted(Comparator.reverseOrder())
    .forEach(System.out::println);
```

Program output.

```
10
8
6
4
2
```

3. Conclusion

In this tutorial, we learned to use the *forEach()* method to iterate through all the elements of a *Stream*.

Though we can use the *enhanced for-each loop* for the iteration, the primary **difference between the forEach() method and for-each loop** is that the ***for-each loop* is an external iterator**, whereas the new ***forEach()* method is an internal iterator**.

Drop me your questions related to **Stream forEach()** method in Java Stream API.

Happy Learning !!

Sourcecode on Github

Was this post helpful?

Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. Java 8 forEach()
2. ArrayList forEach() example – Java 8
3. Java Stream reuse – traverse stream multiple times?
4. Java Stream forEachOrdered()
5. Java Stream min()
6. Java Stream map()
7. Java Stream skip()
8. Java Stream toArray()
9. Java Stream findFirst()
0. Java Stream findAny()

Leave a Comment

Name *

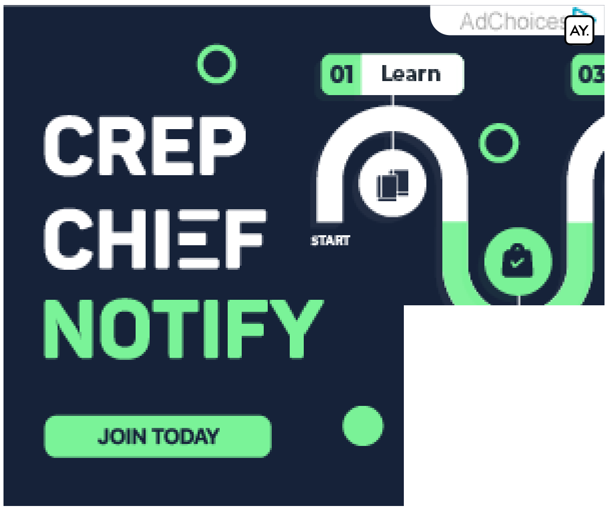
Email *

Website

☐ Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search …


AdChoices
01 Learn
03
CREP
CHIEF
NOTIFY
START
JOIN TODAY

AdChoices
Meet the New SimpliSafe Wireless
Outdoor Security Camera
Home security systems that
choose, set up and
SimpliSafe

AdChoices
Meet the New SimpliSafe Wireless
Outdoor Security Camera
Home security systems that
choose, set up and
SimpliSafe