HowToDoInJava

Creating Streams in Java

Last Updated: February 26, 2022 By: Lokesh Gupta Java Streams Java Stream Basics

Learn to **create streams** of primitives and objects in Java using some most popular ways. We will learn to **create finite as well as infinite streams**.

Table Of Contents

1. Creating Finite Streams

1.1. Empty Stream

We can use `Stream.empty()` method to create an empty stream.

```
Stream<String> emptyStream = Stream.empty();
```

1.2. From Values

In Java, the `Stream.of()` creates a stream of **the supplied values as *var-args*, array or list**.

```
static <T> Stream<T> of(T... values);
```

Let us see a few examples to create a stream of values.

```
Stream<Integer> stream = Stream.of(1,2,3,4,5,6,7,8,9);  //from var args

Stream<Integer> stream = Stream.of( new Integer[]{1,2,3,4,5,6,7,8,9} );  //from array

Employee[] arrayOfEmps = {
    new Employee(1, "A", LocalDate.of(1991, 1, 1), 10000d),
    new Employee(2, "B", LocalDate.of(1992, 1, 1), 20000d),
    new Employee(3, "C", LocalDate.of(1993, 1, 1), 30000d)
};

Stream<Employee> employeeStream = Stream.of(arrayOfEmps);
```

1.3. From Collections

We can also get the **stream from Java collection classes** such as *List*, *Map* and *Set*.

```
List<String> list = Arrays.asList("A", "B", "C", "D");
Stream<String> stream = list.stream();
```

Similarly, get a **stream from Map**.

```
Map<String, Integer> map = new HashMap<>();
map.put("A", 1);

Stream<String> keyStream = map.keySet().stream();
Stream<Integer> valStream = map.values().stream();
Stream<Map.Entry<String, Integer>> entryStream = map.entrySet().stream();
```

We can also **get the stream using utility classes such as *Arrays* and *Collections***.

```
String[] arr = { "A", "B", "C", "D" };

Stream<String> stream = Arrays.stream(arr);
```

1.4. Stream.Builder

The *Stream.Builder* class follows the builder pattern where we add items to the stream in steps, and finally call the method *build()* to get the stream.

```
Stream<String> streamBuilder = Stream.<String>builder()
                                        .add("A")
                                        .add("B")
                                        .build();
```

2. Creating Infinite Streams

Use the following methods to create infinite streams in Java.

- `iterate(seed, function)` – accepts two parameters – a *seed* which is the first term in the stream, and a *function* to produce the value of the next item in the stream. We can limit the stream using the `limit()` method.
- `generate(supplier)` – accepts a `Supplier` that provides **an infinite series of elements** which are placed in the stream. The `limit()` method can then be called in the stream chain to stop the series after a certain number of elements. This is **suitable for generating constant streams, streams of random elements**, etc.

2.1. Stream.iterate()

An example is to generate an infinite stream of even numbers starting from 0 using the *iterate()* function.

```
Stream<Integer> infiniteEvenNumbers = Stream.iterate(0, n -> n + 2).limit(10);
```

2.2. Stream.generate()

A similar example creates a stream of 10 random numbers between 0 and 99 using *generate()* function.

```
Random rand = new Random();

Stream<Integer> stream =
    Stream.generate(() -> rand.nextInt(100)).limit(20);
```

3. Conclusion

In this Java 8 stream tutorial, we learned to **finite stream elements** as well as infinite streams of elements. We saw the usage of `limit()` function which is used to pick the first N elements from an infinite stream.

Happy Learning !!

Sourcecode on Github

Recommended Reading:

1. Creating Infinite Streams in Java
2. Java Streams API
3. Primitive Type Streams in Java
4. Boxed Streams in Java
5. Using 'if-else' Conditions with Java Streams
6. Sorting Streams in Java
7. Applying Multiple Conditions on Java Streams
8. Finding Max and Min from List using Streams
9. Creating an Immutable Class in Java
0. Creating a Temporary File in Java

Leave a Comment

Name *

Email *

Website

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

Post Comment

Search …