HowToDoInJava

Convert between Stream and Array

Last Updated: March 14, 2022 By: Lokesh Gupta Java Array Java Array

Learn to **convert a stream to an array and vice versa** in Java. We will learn to convert for the primitives as well as the Object types.

Quick Reference

```
String[] stringArray = {"a", "b", "c", "d", "e"};

// array -> stream
Stream<String> strStream = Arrays.stream(stringArray);

// stream -> array
String[] stringArray = stream.toArray(String[]::new);
```

Table Of Contents

Note that Java Stream API provides the following **specialized classes for the *stream of primitives***. These classes support many useful sequential and parallel aggregate operations such as `sum()` and `average()`. Consider using these classes to store a stream of primitives for better compatibility with other APIs.

- *IntStream* – Stream of `int` values
- *LongStream* – Stream of `long` values
- *DoubleStream* – Stream of `double` values

1. Converting Array to Stream

1.1. Method Syntax

The primary method to convert an array to a stream of elements is **Arrays.stream()**. It is an overloaded method.

- `Stream<T> stream(T[] array)` : returns a sequential Stream with the specified array as its source.
- `Stream<T> stream(T[] array, int start, int end)` : returns a sequential Stream of array items from index positions `start` (*inclusive*) to `end` (*exclusive*).

Let's under its usage with the following examples.

1.2. Primitive Array to Stream

Java Program to **convert int array to IntStream**.

```
int[] primitiveArray = {0,1,2,3,4};

IntStream intStream = Arrays.stream(primitiveArray);
```

Java Program to **convert int array to Stream of Integer** objects.

```
int[] primitiveArray = {0,1,2,3,4};

Stream<Integer> integerStream = Arrays.stream(primitiveArray)
                                      .boxed();
```

1.3. Object Array to Stream

Java program to **convert an object array to a** stream of objects. We can apply this approach to any type of object, including Java objects (*String*, *Integer* etc.) or custom objects (*User*, *Employee* etc.).

```
String[] stringArray = {"a", "b", "c", "d", "e"};

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

2. Converting Stream to Array

2.1. Method Syntax

The primary method for converting a stream to an array is **Stream.toArray()**. It is also an overloaded method.

- `Object[] toArray()` : returns an array containing the elements of a specified stream. By default, the return type of this method is *Object[]*.
- `T[] toArray(IntFunction<T[]> generator)` : returns an array containing the elements of this stream, using the provided ***generator* function**. The `generator` produces a new array of the desired type and the provided length.

Let us understand the usage of *toArray()* method with some examples.

2.2. Stream to Primitive Array

Java program to **get a stream of ints from `IntStream`**.

```
IntStream intStream = Arrays.stream(new int[]{1,2,3});

int[] primitiveArray = intStream.toArray();
```

Java program to **convert a stream of Integers to primitive int array**. Note that `mapToInt()` returns an instance of *IntStream* type. And `IntStream.toArray()` returns an `int[]`. This is the reason we are not using any *generator* function.

```
Stream<Integer> integerStream = Arrays.stream(new Integer[]{1,2,3});

int[] primitiveArray = integerStream.mapToInt(i -> i).toArray();
```

2.3. Stream to Object Array

Java program to **convert a stream of objects to an array of objects**. It applies to all Java classes and custom objects as well. By default, *toArray()* will return an `Object[]`. To get the `String[]`, we are using the generator function `String[]::new` that creates an instance of *String* array.

```
Stream<String> strStream = Arrays.stream(new String[]{});

String[] stringArray = strStream.toArray(String[]::new);
```

3. Conclusion

In this short tutorial, we learned to convert the stream of items to the array of items, including primitives and complex object types. We learned to use the *Arrays.stream()* and *Stream.toArray()* methods and their examples.

We also learned that it is generally recommended to use specialized classes such as `IntStream` for having the stream of primitive values. These classes provide custom methods for primitive types, and many helpful utility methods.

Happy Learning !!

Sourcecode on Github

Recommended Reading:

1. Convert Between Array of Primitives and Array to Objects
2. Convert byte[] Array to String and Vice-versa
3. Convert List to Array and Vice versa
4. GSON – Parse JSON array to Java array or list
5. Get all Dates between Two Dates as Stream
6. Collecting Stream of Primitives into Collection or Array
7. Convert between LocalDateTime and ZonedDateTime
8. Convert between Date to LocalDateTime
9. Convert between LocalDate to java.sql.Date
0. Convert between LocalDate and LocalDateTime

Leave a Comment

Name *

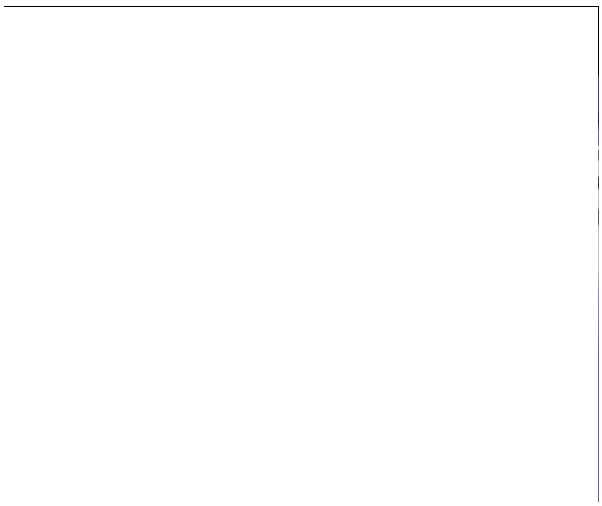
Email *

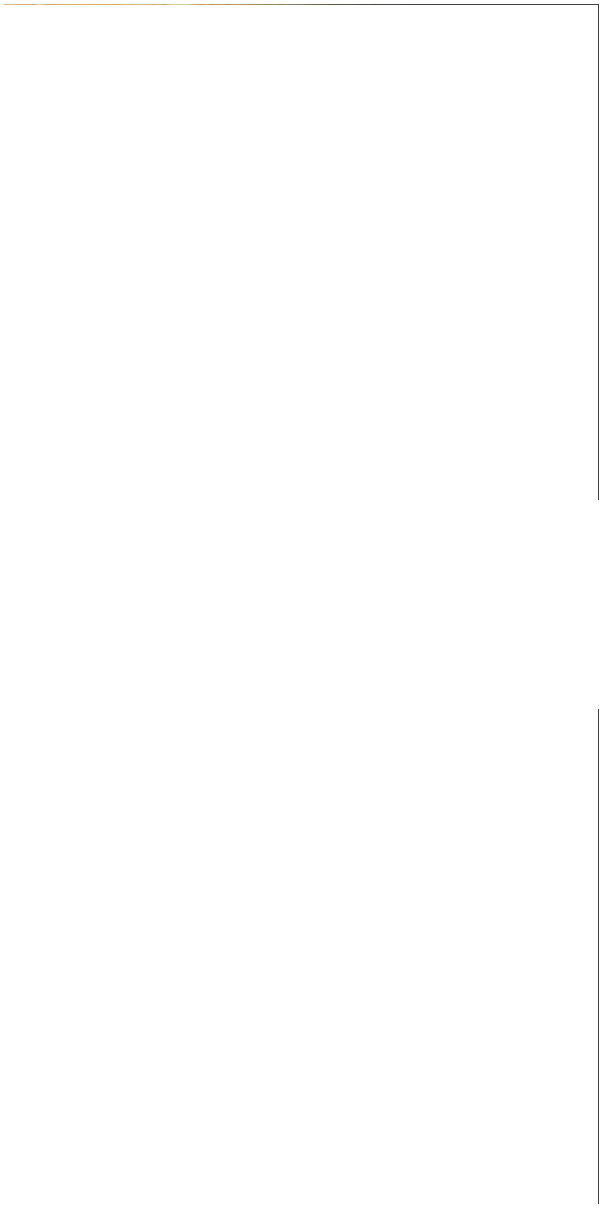
Website

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

Post Comment

Search …





HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise

Guest Posts

Blogs

REST API Tutorial