

# Building a Stream from Data in Memory

---



**José Paumard**

PHD, JAVA CHAMPION, JAVA ROCK STAR

@JosePaumard <https://github.com/JosePaumard>



# Agenda



Let us see some more Stream in action!

Let us create streams

And process them



# Creating Streams

---





## Five patterns to create streams:

- from a collection
- from an array
- from a text file
- from a regular expression
- from a String

```
List<Person> people = ...;
```

```
Stream<Person> peopleStream = people.stream();
```

Creating a stream from a Collection



```
Person[] people = ...;
```

```
Stream<Person> peopleStream1 = Arrays.stream(people);
```

```
Stream<Person> peopleStream2 = Stream.of(people);
```

Creating a stream from an array

Using the factory method `Arrays.stream()`

Or the factory method `Stream.of()`



```
Path path = Path.of("files/data.txt");

try {
    Stream<Person> peopleStream = Files.lines(path)

    // peopleStream is available for map / filter / reduce

} catch (IOException ioe) {
    // do something smart with the exception
}
```

Creating a **path** to a file

Then a stream using the **factory method** `Files.lines()`



```
Path path = Path.of("files/data.txt");

try (Stream<Person> peopleStream = Files.lines(path)) {

    // peopleStream is available for map / filter / reduce

} catch (IOException ioe) {
    // do something smart with the exception
}
```

Creating a path to a file

Then a stream using the factory method `Files.lines()`

- a stream is auto-closeable and will close the reader
- `lines()` can also take a charset in case it is not UTF-8





```
String sentence = "the quick brown fox";  
  
Pattern pattern = Pattern.compile(" ");  
  
Stream<String> words =  
    pattern.splitAsStream(sentence);
```

Creating a pattern on a regular expression to split a String

With the `splitAsStream()` method

The sentence is split one element at a time



```
String word = "supercalifragilisticexpialidocious";  
  
IntStream letters = word.chars();  
  
Stream<String> lettersAsString =  
    letters.mapToObj(Character::toString);
```

Creating a stream of letters on a String

With some conversion to be made to make it a Stream<String>...



# Demo



Let us write some code!

Create these streams



# Selecting Elements of a Stream

---





Two patterns to select elements from a stream, based on:

- their **index**: `skip()` and `limit()`
- a **predicate**: `takeWhile()` and `dropWhile()`

# Demo



Let us write some code!

Let us skip and limit elements

And take them while a predicate is true



# Module Wrap Up



What did you learn?

Streams are not just about collections

`Stream<T>` and `IntStream`

Intermediate operations to control  
streams

