Collectors API

Alan Biju
@itsmeambro

Collectors API

Collectors are tools used with Java Streams to gather the processed data into a final result like a list, set, map, or even a single value. They work with the collect() method to group, filter, count, or join stream elements easily.

1. Collect to List

Before (Java 7):

```
List<String> names = Arrays.asList("Alan", "Bob", "Alice");
List<String> result = new ArrayList<>();
for (String name : names) {
    if (name.startsWith("A")) {
        result.add(name);
    }
}
System.out.println(result);
```

After (Java 8):

```
List<String> names = Arrays.asList("Alan", "Bob", "Alice");
List<String> result = names.stream()
    .filter(name -> name.startsWith("A"))
    .collect(Collectors.toList());
System.out.println(result);
```

2. Collect to Set

Before:

```
List<String> list = Arrays.asList("apple",
          "banana", "apple", "orange");
Set<String> set = new HashSet<>();
for (String s : list) {
    set.add(s);
}
System.out.println(set);
```

After:

```
Set<String> set = list.stream()
    .collect(Collectors.toSet());
System.out.println(set);
```


Alan Biju
@itsmeambro

3. Counting Elements

Before:

```
  List<String> names = Arrays.asList("A", "B", "C", "A");
  int count = 0;
  for (String s : names) {
    if (s.equals("A")) {
        count++;
    }
  }
  System.out.println(count);
```

After:

```
  long count = names.stream()
    .filter(s -> s.equals("A"))
    .count();
System.out.println(count);
```

Or using Collectors.counting():

```
  long count = names.stream()
    .filter(s -> s.equals("A"))
    .collect(Collectors.counting());
```

4. Joining Strings

Before:

```
  List<String> list = Arrays.asList("Java", "Python", "Go");
  StringBuilder sb = new StringBuilder();
  for (String s : list) {
    if (sb.length() > 0) sb.append(", ");
    sb.append(s);
  }
  System.out.println(sb.toString());
```

After:

```
  String result = list.stream()
    .collect(Collectors.joining(", "));
  System.out.println(result);
```


Alan Biju
@itsmeambro

5. Grouping By Field

Before:

```
List<Person> people = Arrays.asList(
  new Person("Alan", "NY"),
  new Person("Bob", "LA"),
  new Person("Alice", "NY")
);
Map<String, List<Person>> map = new HashMap<>();
for (Person p : people) {
  if (!map.containsKey(p.city)) {
      map.put(p.city, new ArrayList<>());
  }
  map.get(p.city).add(p);
}
System.out.println(map);
```

After:

```
Map<String, List<Person>> map = people.stream()
  .collect(Collectors.groupingBy(p -> p.city));
System.out.println(map);
```

6. Partitioning (true/false separation)

Before:

```
List<Integer> nums = Arrays.asList(1, 2, 3, 4, 5);
Map<Boolean, List<Integer>> result = new HashMap<>();
  result.put(true, new ArrayList<>());
  result.put(false, new ArrayList<>());
for (Integer i : nums) {
  if (i % 2 == 0)
      result.get(true).add(i);
  else
      result.get(false).add(i);
}
System.out.println(result);
```

After:

```
Map<Boolean, List<Integer>> result = nums.stream()
      .collect(Collectors.partitioningBy(i -> i % 2 == 0));
System.out.println(result);
```


Alan Biju
@itsmeambro

7. Summing Int

Before:

```
List<Integer> nums = Arrays.asList(1, 2, 3);
int sum = 0;
for (Integer n : nums) {
  sum += n;
}
System.out.println(sum);
```

After:

```
int sum = nums.stream()
  .collect(Collectors.summingInt(i -> i));
System.out.println(sum);
```

8. Mapping with Collectors.mapping()

Before:

```
List<Person> people = Arrays.asList(
  new Person("Alan", "NY"),
  new Person("Bob", "LA")
);

List<String> names = new ArrayList<>();
for (Person p : people) {
  names.add(p.name);
}
System.out.println(names);
```

After:

```
List<String> names = people.stream()
  .collect(Collectors.mapping(p -> p.name, Collectors.toList()));
System.out.println(names);
```

9. Finding Max/Min using Comparator

Before:

```
List<Integer> list = Arrays.asList(3, 5, 7, 2);
int max = Integer.MIN_VALUE;
for (int num : list) {
  if (num > max) {
     max = num;
  }
}
System.out.println(max);
```


Alan Biju
@itsmeambro

After:

```
int max = list.stream()
  .max(Integer::compare)
  .orElseThrow();
System.out.println(max);
```

10. Collecting into Map with Custom Key/Value

Before:

```
List<Person> people = Arrays.asList(
  new Person("Alan", "NY"),
  new Person("Bob", "LA")
);
Map<String, String> map = new HashMap<>();
for (Person p : people) {
  map.put(p.name, p.city);
}
System.out.println(map);
```

After:

```
Map<String, String> map = people.stream()
  .collect(Collectors.toMap(p -> p.name, p -> p.city));
System.out.println(map);
```


Alan Biju
@itsmeambro

If you find this helpful, like and share it with your friends

Alan Biju
@itsmeambro