HowToDoInJava

Collecting Stream Items into Map in Java

Last Updated: March 14, 2022 By: Lokesh Gupta Java 8 Java Stream Basics

Learn to **collect Stream items into Map** using `Collectors.toMap()` and `Collectors.groupingBy()` methods using Java Stream APIs.

Table Of Contents

1. Collectors.toMap() for Unique Key-value Pairs

If the **stream items have the unique map key field** then we can use `Collectors.toMap()` to collect items to Map in `Map<keyObj, Item>` format.

For example, we can collect a list of `Employee` objects to *Map* in where employee ids are unique fields and used as keys to the *Map* entries.

When Map Keys are Unique

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.function.Function;
import java.util.stream.Collectors;

public class Main
{
    public static void main(String[] args)
    {
        List<Employee> employeeList = new ArrayList<>(Arrays.asList(
                                    new Employee(1, "A", 100),
                                    new Employee(2, "A", 200),
                                    new Employee(3, "B", 300),
                                    new Employee(4, "B", 400),
                                    new Employee(5, "C", 500),
                                    new Employee(6, "C", 600)));

        Map<Long, Employee> employeesMap = employeeList.stream()
                                        .collect( Collectors.toMap(Employee::
                                                Function.identity()) );

        System.out.println(employeesMap);
    }
}
```

Program output.

Output

```
{1=Employee [id=1, name=A, salary=100.0],
2=Employee [id=2, name=A, salary=200.0],
3=Employee [id=3, name=B, salary=300.0],
4=Employee [id=4, name=B, salary=400.0],
5=Employee [id=5, name=C, salary=500.0],
6=Employee [id=6, name=C, salary=600.0]}
```

2. Collectors.groupingBy() when Multiple Keys have Same Value

If the stream has items where Map keys are duplicate then we can use Collectors.groupingBy() to collect elements in `Map<key, List<value>>` format. Here for each map key, we will store all elements in a *List* as the value.

For example, we can collect a list of `Employee` objects to map in where employee names may be duplicate fields for some stream elements. In such a case, all employees with the same name will be stored in a *List*, and the list will be stored as *Map* value field.

When Map Keys are Duplicate

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

public class Main
{
    public static void main(String[] args)
    {
        List<Employee> employeeList = new ArrayList<>(Arrays.asList(
                                new Employee(1, "A", 100),
                                new Employee(2, "A", 200),
                                new Employee(3, "B", 300),
                                new Employee(4, "B", 400),
                                new Employee(5, "C", 500),
                                new Employee(6, "C", 600)));

        Map<String, List<Employee>> employeesMap = employeeList.strea
                                    .collect(Collectors.groupingBy(Employ

        System.out.println(employeesMap);
    }
}
```

Program output.

Output

```
{A=[Employee [id=1, name=A, salary=100.0], Employee [id=2, name=A, sa
 B=[Employee [id=3, name=B, salary=300.0], Employee [id=4, name=B, sa
 C=[Employee [id=5, name=C, salary=500.0], Employee [id=6, name=C, sa
```

3. Conclusion

It is very important to know beforehand if the `Stream` elements will have a distinct value for the map key field or not.

If map keys are duplicate and we use `Collectors.toMap()` method, we will get the **IllegalStateException**:

Error

```
Exception in thread "main" java.lang.IllegalStateException: Duplicate
    at java.util.stream.Collectors.lambda$throwingMerger$106(Collecto
    at java.util.stream.Collectors$$Lambda$3/149928006.apply(Unknown
    at java.util.HashMap.merge(HashMap.java:1245)
```

Happy Learning !!

Sourcecode on Github

Was this post helpful?

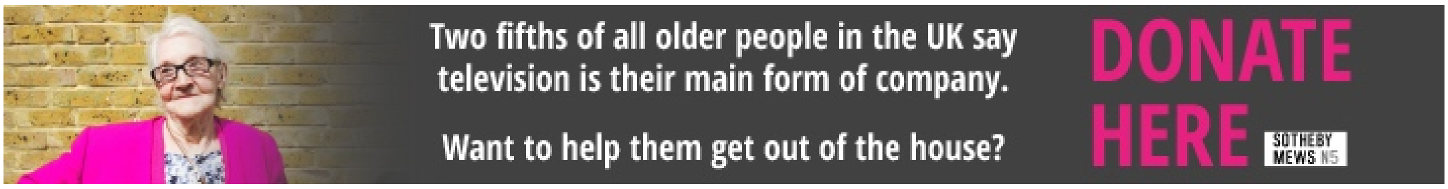
Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. Collecting Stream Items into List in Java
2. Collecting Stream of Primitives into Collection or Array
3. Jackson – Convert JSON to Map and Map to JSON
4. Getting Distinct Stream Items by Comparing Multiple Fields
5. Append or Prepend Items to a Stream
6. Java Stream map()
7. Java Stream map() vs flatMap()
8. Java Stream reuse – traverse stream multiple times?
9. Removing Items from an Array in Java
0. Finding Top N Items in Array


Two fifths of all older people in the UK say television is their main form of company.
Want to help them get out of the house?
DONATE HERE
SOTHEBY MEWS N5

** We do not spam !!*

2 thoughts on “Collecting Stream Items into Map in Java”

Buc

November 6, 2019 at 5:51 pm

If you wish to create map from list by id, but list contains duplicates you can use Collectors.toMap with BinnaryOperator:

```
public static void main (String[] args) {
  List<Employee> employeeList = Arrays.asList(
                new Employee(1, "A", 100),
                new Employee(1, "A", 100),
                new Employee(2, "A", 200),
                new Employee(3, "B", 300),
                new Employee(4, "B", 400),
                new Employee(5, "C", 500),
                new Employee(5, "C", 500),
                new Employee(6, "C", 600));


        Map<Long, Employee> employeesMap = employeeList.stream()
```

```
                    .collect(Collectors.toMap(Employee::getId,
                            Function.identity(), (first, second) -> first));

            employeesMap.entrySet().forEach(System.out::println);
    }
```

The console output will be:

```
1=Employee[id=1, name='A', salary=100.0]
2=Employee[id=2, name='A', salary=200.0]
3=Employee[id=3, name='B', salary=300.0]
4=Employee[id=4, name='B', salary=400.0]
5=Employee[id=5, name='C', salary=500.0]
6=Employee[id=6, name='C', salary=600.0]
```

Reply

Lokesh Gupta

November 6, 2019 at 11:01 pm

Thanks for sharing.

Reply

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