HowToDoInJava

Getting Distinct Stream Items by Comparing Multiple Fields

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

Learn to *collect or count distinct objects from a stream* where each object is **distinct by comparing multiple fields** in the class.

Java does not have direct support for finding such distinct items from the Stream where items should be distinct by multiple fields. So, we will create a custom *Predicate* for this purpose.

Table Of Contents

1. Finding Distinct Items by Multiple Fields

Below given is a function that accepts **varargs** parameters and returns a *Predicate* instance. We can use this function to pass multiple **key extractors** (fields on which we want to filter the duplicates).

This function creates a *List* of field values and this *List* act as a single key for that *Stream* item. The *list* contains the values of fields to check distinct combinations.

Then these keys are inserted into a `ConcurrentHashMap` that allows only unique keys.

Predicate distinctByKeys()

```
private static <T> Predicate<T>
    distinctByKeys(final Function<? super T, ?>... keyExtractors)
{
    final Map<List<?>, Boolean> seen = new ConcurrentHashMap<>();

    return t ->
    {
      final List<?> keys = Arrays.stream(keyExtractors)
                  .map(ke -> ke.apply(t))
                  .collect(Collectors.toList());

      return seen.putIfAbsent(keys, Boolean.TRUE) == null;
    };
}
```

In the given example, we are finding all persons having distinct ids and names. We should have only 3 records as output.

Demo

```
Collection<Person> list = Arrays.asList(alex, brianOne,
        brianTwo, lokeshOne,
        lokeshTwo, lokeshThree);

List<Person> distinctPersons = list.stream()
      .filter(distinctByKeys(Person::firstName, Person::lastName))
      .collect(Collectors.toList());
```

Here *Person* may be a class or record.

```
record Person(Integer id, String firstName, String lastName, String email) {
}
```

2. Distinct by Multiple Fields using Custom Key Class

Another possible approach is to have **a custom class that represents the distinct key** for the POJO class.

For the previous example, we can create a class **CustomKey** containing id and name values. The distinct elements from a list will be taken based on the distinct combination of values for all these fields.

In the given example, again, we are finding all records having unique ids and names. Note that in this approach, **we are only replacing the List with CustomKey** class.

CustomKey.java

```
record CustomKey(String firstName, String lastName) {
  public CustomKey(final Person p)
  {
    this(p.firstName(), p.lastName());
  }
}
```

Let us see how `CustomKey::new` is used for filtering the distinct elements from the list based on the given multiple fields.

Demo

```
Collection<Person> list = Arrays.asList(alex, brianOne,
    brianTwo, lokeshOne,
    lokeshTwo, lokeshThree);

List<Person> distinctPersons = list.stream()
```

```
        .filter(distinctByKeyClass(CustomKey::new))
        .collect(Collectors.toList());

//Method accepting Custom key class
public static <T> Predicate<T>
    distinctByKeyClass(final Function<? super T, Object> keyExtractor)
{
    Map<Object, Boolean> seen = new ConcurrentHashMap<>();
    return t -> seen.putIfAbsent(keyExtractor.apply(t), Boolean.TRUE) == null;
}
```

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. Sorting a Stream by Multiple Fields in Java
2. Java Stream distinct()
3. Java Stream reuse – traverse stream multiple times?
4. Collecting Stream Items into Map in Java
5. Collecting Stream Items into List in Java
6. Append or Prepend Items to a Stream
7. Adding Multiple Items to ArrayList
8. Gson – Exclude or Ignore Fields
9. Getting the Last Item of a Stream
0. Comparing SOAP vs REST APIs

3 thoughts on "Getting Distinct Stream Items by Comparing Multiple Fields"

Robert

May 21, 2020 at 7:53 am

How can we get same list of Record with all distinct count as comma seperated? consider count is String field Record [id=1, count=[10,20], name=record1, email=record1@email.com, location=India],

Reply

Gerald Butler

August 10, 2019 at 7:55 pm

I noticed that both of the above solutions will allocate a new object on every single element of the stream. For a large stream, that will be a lot of garbage created that needs to be collected. Here is a solution I came up with that avoids allocation on every single element of the stream. Seems to work welll:

```
public final class StreamHelpers {

    private StreamHelpers() {
    }

    @SuppressWarnings("unchecked")
    public static <T> Function<T, Stream<T>> distinctBy(Function<T, ?>... fieldSelectors) {
        ConcurrentMap keysSeen = new ConcurrentHashMap<>();
        return t -> {
            ConcurrentMap keyMap = keysSeen;
            for (int i = 0; i < fieldSelectors.length - 1; i++) {
                Function<T, ?> selector = fieldSelectors[i];
                keyMap = (ConcurrentMap) keyMap.computeIfAbsent
                        (
                                selector.apply(t),
                                k -> new ConcurrentHashMap()
                        );
            }
            boolean seen = (boolean) keyMap.compute
                    (
                            fieldSelectors[fieldSelectors.length - 1].apply(t),
                            (k, v) -> v != null
                    );
            return seen ? null : Stream.of(t);
        };
    }
}
```

It is used like:

```
foos.stream().flatMap(distinctBy(Foo::getKey1, Foo::getKey2, ... Foo::getKeyN))
```

Unfortunately, it still allocates once for each element the first time that element's key is seen (Stream.of(…)). However, if there are a significant number of duplicates, this should reduce the allocations significantly. It would be nice to have a solution that involved zero allocations if possible though. I made it work using flatMap instead of filter because I felt having a Predicate that has side-effects violates the contract of filter. I was trying to get something that follows all the contracts of the stream methods AND avoids all allocations (or at least minimizes them). Any ideas for how to get there better?

Reply

Amirul Syafi

April 12, 2019 at 2:11 pm

Hello ,

i followed your tutorial , right now it showing my desired output, but still i need to get the total count of the filtered item and put it in object list -> so i can show how many count from each group

Hope you can help me , Thanks

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 ...