# Java利用stream(流)对map中的values进行过滤、排序操作



*前言*

> 对于Java8中的stream(流)这种优雅、方便、快捷、高效的操作已经是信手沾来了吧，但是却仅限List、Set。难道我Map不要面子得嘛？在工作中需要对从List转Map的数据进行操作，因此便有这随笔。

新建一个`User`类，注意下面使用`lombok`；如果你是`idea`请下载相关插件以及依赖，我这里使用`maven`引，在这里不多详细说明了。

Copy

```
/**
 * @program: strategy-demo
 * @description: stream操作
 * @author: YuAoTian
 * @create: 2020-01-06 23:23
 **/
@Data
@AllArgsConstructor
class User{
    private Integer id;
    private String name;
    private String city;
    private Integer age;
}
```

---

继续看代码，看`Main`方法里面的

Copy

```
 public static void main(String[] args) {
        List<User> userList = Arrays.asList(
                new User((int) (Math.random()*10),"小明","北京",18),
                new User((int) (Math.random()*10),"小红", "南京", 17),
                new User((int) (Math.random()*10),"小丫", "湖北", 14),
                new User((int) (Math.random()*10),"小黑", "深圳", 22),
                new User((int) (Math.random()*10),"小玉", "广州", 19),
                new User((int) (Math.random()*10),"小小", "江西", 21)
                );
      Map<Integer, List<User>> userMap =
userList.stream().collect(Collectors.groupingBy(User::getId));//已ID作为Key
    }
```

## Map排序#

### 正排#

Copy

```
Map<Integer, List<User>> map =
userMap.entrySet().stream().sorted(Comparator.comparing(o ->
o.getValue().get(0).getAge())).map(entry -> {
            Map<Integer, List<User>> result = new LinkedHashMap<>();
            result.put(entry.getKey(), entry.getValue());
            return result;
        }).reduce((map1, map2) -> {
            map2.forEach(map1::put);
            return map1;
        }).get();
         System.out.print(map);
```

*输出*

Copy

```
{5=[User(id=5, name=小丫, city=湖北, age=14), User(id=5, name=小玉, city=广州,
age=19), User(id=5, name=小小, city=江西, age=21)], 0=[User(id=0, name=小红, city=南
京, age=17)], 8=[User(id=8, name=小明, city=北京, age=18)], 9=[User(id=9, name=小黑,
city=深圳, age=22)]}
```

## 倒排#

Copy

```
 Map<Integer, List<User>> map =
userMap.entrySet().stream().sorted(Comparator.comparing(o -> {
            //倒排中 reversed() 方法 是object 对象，需要在里面强制回 Entry 就行。
            Map.Entry<Integer, List<User>> temp = (Map.Entry<Integer, List<User>>)
o;
            return temp.getValue().get(0).getAge();
        }).reversed()).map(entry -> {
            Map<Integer, List<User>> result = new LinkedHashMap<>();
            result.put(entry.getKey(), entry.getValue());
            return result;
        }).reduce((map1, map2) -> {
            map2.forEach(map1::put);
            return map1;
        }).get();
        System.out.print(map);
```

*输出*

Copy

```
{3=[User(id=3, name=小黑, city=深圳, age=22)], 5=[User(id=5, name=小玉, city=广州,
age=19)], 0=[User(id=0, name=小明, city=北京, age=18), User(id=0, name=小丫, city=湖
北, age=14)], 2=[User(id=2, name=小红, city=南京, age=17), User(id=2, name=小小,
city=江西, age=21)]}
```

以上是对年龄进行从小到大和从大到小的排序，并且最后返回的是一个`Optional`，如果不懂的可以接下来看一些例子。

Copy

```
Optional<Map<Integer, List<User>>> map =
userMap.entrySet().stream().sorted(Comparator.comparing(o -> {
            //倒排中 reversed() 方法 是object 对象，需要在里面强制回 Entry 就行。
            Map.Entry<Integer, List<User>> temp = (Map.Entry<Integer, List<User>>)
o;
            return temp.getValue().get(0).getAge();
        }).reversed()).map(entry -> {
            Map<Integer, List<User>> result = new LinkedHashMap<>();
            result.put(entry.getKey(), entry.getValue());
            return result;
        }).reduce((map1, map2) -> {
            map2.forEach(map1::put);
            return map1;
        });
                //为null返回false，否则true，即可调用get()返回~
        if(map.isPresent()){
            Map<Integer, List<User>> listMap = map.get();//调用get()返回一个Map
        }
                System.out.print(map);
```

### 优化#

Copy

```
Map<Integer, List<User>> map =
userMap.entrySet().stream().sorted(Comparator.comparing(o -> {
            //倒排中 reversed() 方法 是object 对象，需要在里面强制回 Entry 就行。
            Map.Entry<Integer, List<User>> temp = (Map.Entry<Integer, List<User>>)
o;
            return temp.getValue().get(0).getAge();
        }).reversed()).collect(LinkedHashMap::new,(m,e) ->
m.put(e.getKey(),e.getValue()),LinkedHashMap::putAll);
```

是不是瞬间感觉世界清静了很多~你还可以处理一下一些缩进，能把他缩到一行！

---

## Map过滤#

Copy

```
map.entrySet().stream().filter(entry -> entry.getValue().get(0).getAge() >
16).collect(LinkedHashMap::new, (m, e) -> m.put(e.getKey(), e.getValue()),
LinkedHashMap::putAll);
 System.out.println(map);
```

*输出*

Copy

```
{1=[User(id=1, name=小丫, city=湖北, age=14)]}
```

过滤掉年龄`<`16的，过滤非常简单我就不提供其他例子了；一个filter然后直接collect就行了~

---