

02_commons-collections4#CollectionUtils

判空、null处理

包含

集合运算（和、差、交）

►附录一：集合运算（和、差、交）

创建时间： 2023年08月17日 16:43:05

提示：

- 虽然dubbo、spring框架都有同名工具类，但是那都是方便它们框架内部使用的工具😞
 - 然而，commons-collections4中的CollectionUtils具有丰富且强大的集合操作API😄
- 😊集合相关的运算应该第一时间想到commons-collections4中的 `union` + `removeAll` + `retainAll` !

判空、null处理 为null时设置空集合、列表、Set

- 判空
 - isEmpty
 - isEmpty

```
CollectionUtils.isEmpty_
  m isEmpty(Collection<?> coll)
  m isEmpty(Collection<?> coll)
```

- null处理 ⚠不同的工具会返回不同类型的结果!
 - CollectionUtils#emptyIfNull
 - ListUtils#emptyIfNull + ListUtils#DefaultIfNull
 - SetUtils#emptyIfNull
 - MapUtils#emptyIfNull

包含

- containsAll

- containsAny ➡ 用于在DB存在联合唯一索引，添加数据时判重很方便

CollectionUtils.conta

```
(m) containsAll(Collection<?> coll1, Collection<?> coll2)
(m) containsAny(Collection<?> coll1, T... coll2)
(m) containsAny(Collection<?> coll1, Collection<?> coll2)
```

按 Ctrl+. 选择所选 (或第一个) 建议，然后插入点 下一提示

集合运算（和、差、交） Java代码示例见附录一

😄如果操作数据是List，推荐直接使用ListUtils，⚠注意，ListUtils中不包含addAll，这也是推荐使用接口自身addAll的原因。

- 😄集合相关的运算应该第一时间想到commons-collections4中的 `union` + `removeAll` + `retainAll` !
- `union`
- addAll ⚠和Collection接口一样，会修改原数据！请使用Collection接口默认的addAll！
- `removeAll`
Collection接口的removeAll会修改原数据，CollectionUtils中的removeAll不会修改原数据（将返回新数据）
- subtract 只会移除匹配到的第一个元素，适合Set等不含重复元素的集合
- `retainAll`

```

1  @Test
2  public void test01() {
3      List<Integer> integerList = Lists.newArrayList(777, 888, 999);
4      CollectionUtils.addAll(integerList, 1, 2, 3, 4, 5, 5, 5);
5      List<Integer> integerList1 = Lists.newArrayList();
6      CollectionUtils.addAll(integerList1, 1, 2, 3, 4, 5, 6, 7, 8, 99999, 99999, 99999);
7
8      // A+B
9      // → 返回新数据（无序合并）
10     Collection<Integer> unionResult = CollectionUtils.union(
11         CollectionUtils.emptyIfNull(integerList),
12         CollectionUtils.emptyIfNull(integerList1)
13     );
14
15     // A+B
16     // → 会修改原数据（有序合并）
17     boolean addAllBooleanResult = CollectionUtils.addAll(
18         CollectionUtils.emptyIfNull(integerList),
19         CollectionUtils.emptyIfNull(integerList1)
20     );
21
22
23     // A-B
24     // → 移除A中所有在B中存在的元素（移除所有）
25     Collection<Integer> removeAllResult = CollectionUtils.removeAll(
26         CollectionUtils.emptyIfNull(integerList),
27         CollectionUtils.emptyIfNull(integerList1)
28     );
29
30     // A-B
31     // → 移除A中在B中存在的元素（移除一个）
32     Collection<Integer> subtractResult = CollectionUtils.subtract(
33         CollectionUtils.emptyIfNull(integerList),
34         CollectionUtils.emptyIfNull(integerList1)
35     );
36     System.out.println("subtractResult = " + subtractResult);
37
38     // A∩B
39     Collection<Integer> retainAllResult = CollectionUtils.retainAll(
40         CollectionUtils.emptyIfNull(integerList),
41         CollectionUtils.emptyIfNull(integerList1)
42     );
43     // commons-collection4
44     // Set<Integer> retainAllSetResult = SetUtils.hashSet(retainAllResult.toArray(new Integer[0]));
45     // guava
46     // Set<Integer> retainAllSetResult = Sets.newHashSet(retainAllResult.toArray(new Integer[0]));
47
48 }

```

► 附录一：集合运算（和、差、交）

```

1  @Test
2  public void test01() {
3      List<Integer> integerList = Lists.newArrayList(777, 888, 999);
4      CollectionUtils.addAll(integerList, 1, 2, 3, 4, 5, 5, 5);
5      List<Integer> integerList1 = Lists.newArrayList();
6      CollectionUtils.addAll(integerList1, 1, 2, 3, 4, 5, 6, 7, 8, 99999, 99
999, 99999);
7
8      // A+B
9      // → 返回新数据（无序合并）
10     Collection<Integer> unionResult = CollectionUtils.union(
11         CollectionUtils.emptyIfNull(integerList),
12         CollectionUtils.emptyIfNull(integerList1)
13     );
14
15     // A+B
16     // → 会修改原数据（有序合并）
17     boolean addAllBooleanResult = CollectionUtils.addAll(
18         CollectionUtils.emptyIfNull(integerList),
19         CollectionUtils.emptyIfNull(integerList1)
20     );
21
22
23     // A-B
24     // → 移除A中所有在B中存在的元素（移除所有）
25     Collection<Integer> removeAllResult = CollectionUtils.removeAll(
26         CollectionUtils.emptyIfNull(integerList),
27         CollectionUtils.emptyIfNull(integerList1)
28     );
29     // A-B
30     // → 移除A中在B中存在的元素（移除一个）
31     Collection<Integer> subtractResult = CollectionUtils.subtract(
32         CollectionUtils.emptyIfNull(integerList),
33         CollectionUtils.emptyIfNull(integerList1)
34     );
35     System.out.println("subtractResult = " + subtractResult);
36
37
38     // A∩B
39     Collection<Integer> retainAllResult = CollectionUtils.retainAll(
40         CollectionUtils.emptyIfNull(integerList),
41         CollectionUtils.emptyIfNull(integerList1)
42     );
43     // commons-collection4
44     // Set<Integer> retainAllSetResult = SetUtils.hashSet(retainAllResult.t
oArray(new Integer[0]));
45     // guava

```

```
46         // Set<Integer> retainAllSetResult = Sets.newHashSet(retainAllResult.toArray(new Integer[0]));
47
48     }
```

END.