Класс CollectionUtils из Apache Commons

JSP & Servlets
20 уровень, 4 лекция

ОТКРЫТА

Знакомство с CollectionUtils

Еще один универсальный утилитный класс, который содержит много полезных методов:

`addAll(Collection<C> collection, C... elements)`	Добавляет все элементы массива C в коллекцию
`addIgnoreNull(Collection<T> collection, T object)`	Добавляет элемент, если он не null
`containsAll(Collection<?> coll1, Collection<?> coll2)`	Проверяет, что коллекция1 содержит все элементы коллекции2
`containsAny(Collection<?> coll1, Collection<?> coll2)`	Проверяет, что коллекция1 содержит хотя бы один элемент из коллекции2
`countMatches(Iterable<C> input, Predicate<? super C> predicate)`	Подсчитывает, сколько элементов коллекции удовлетворяют правило predicate
`disjunction(Iterable<? extends O> a, Iterable<? extends O> b)`	Возвращает объекты, которые входят только в одну коллекцию, но не сразу в обе
`intersection(Iterable<? extends O> a, Iterable<? extends O> b)`	Возвращает коллекцию объектов, которые входят сразу в обе коллекции
`union(Iterable<? extends O> a, Iterable<? extends O> b)`	Возвращает коллекцию объектов, которые входят хотя бы в одну коллекцию
`emptyCollection()`	Возвращает специальную пустую коллекцию
`emptyIfNull(Collection<T> collection)`	Возвращает пустую коллекцию, если collection == null
`exists(Iterable<C> input, Predicate<? super C> predicate)`	Проверяет, что в коллекции существует элемент, удовлетворяющий правилу predicate
`extractSingleton(Collection<E> collection)`	Возвращает единственный элемент коллекции
`filter(Iterable<T> collection, Predicate<? super T> predicate)`	Фильтрует элементы коллекции
`find(Iterable<T> collection, Predicate<? super T> predicate)`	Ищет элементы коллекции
`isEmpty(Collection<?> coll)`	Проверяет, что коллекция пуста

`isEqualCollection(Collection<?> a, Collection<?> b)`	Сравнивает коллекции
`isFull(Collection<? extends Object> coll)`	Проверяет, можно ли еще добавить элемент в коллекцию
`isNotEmpty(Collection<?> coll)`	Проверят, что коллекция не пуста
`isSubCollection(Collection<?> a, Collection<?> b)`	Проверяет, что коллекция B входит в коллекцию A
`matchesAll(Iterable<C> input, Predicate<? super C> predicate)`	Проверяет, что все элементы коллекции удовлетворяют правилу predicate
`removeAll(Collection<E> collection, Collection<?> remove)`	Удаляет все элементы коллекции
`retainAll(Collection<C> collection, Collection<?> retain)`	Возвращает коллекцию, содержащую все элементы collection, которые также содержатся в retain
`reverseArray(Object[] array)`	Разворачивает массив задом на перед
`unmodifiableCollection(Collection<? extends C> collection)`	Возвращает обертку над коллекцией. Любая попытка модифицировать коллекцию кинет исключение.

Рассмотрим некоторые методы.

Способы проверить пустая ли коллекция

Вот как просто определить, является ли коллекция пустой:

```
CollectionUtils.isEmpty(null); // true
CollectionUtils.isEmpty(new LinkedList<>()); // true
CollectionUtils.isEmpty(Collections.singleton(new int[]{2, 1})); // false
```

Так же легко проверить, что коллекция не пустая:

```
CollectionUtils.isNotEmpty(null); // false
CollectionUtils.isNotEmpty(new LinkedList<>()); // false
CollectionUtils.isNotEmpty(Collections.singleton(new int[]{2, 1})); // true
```

Примеры:

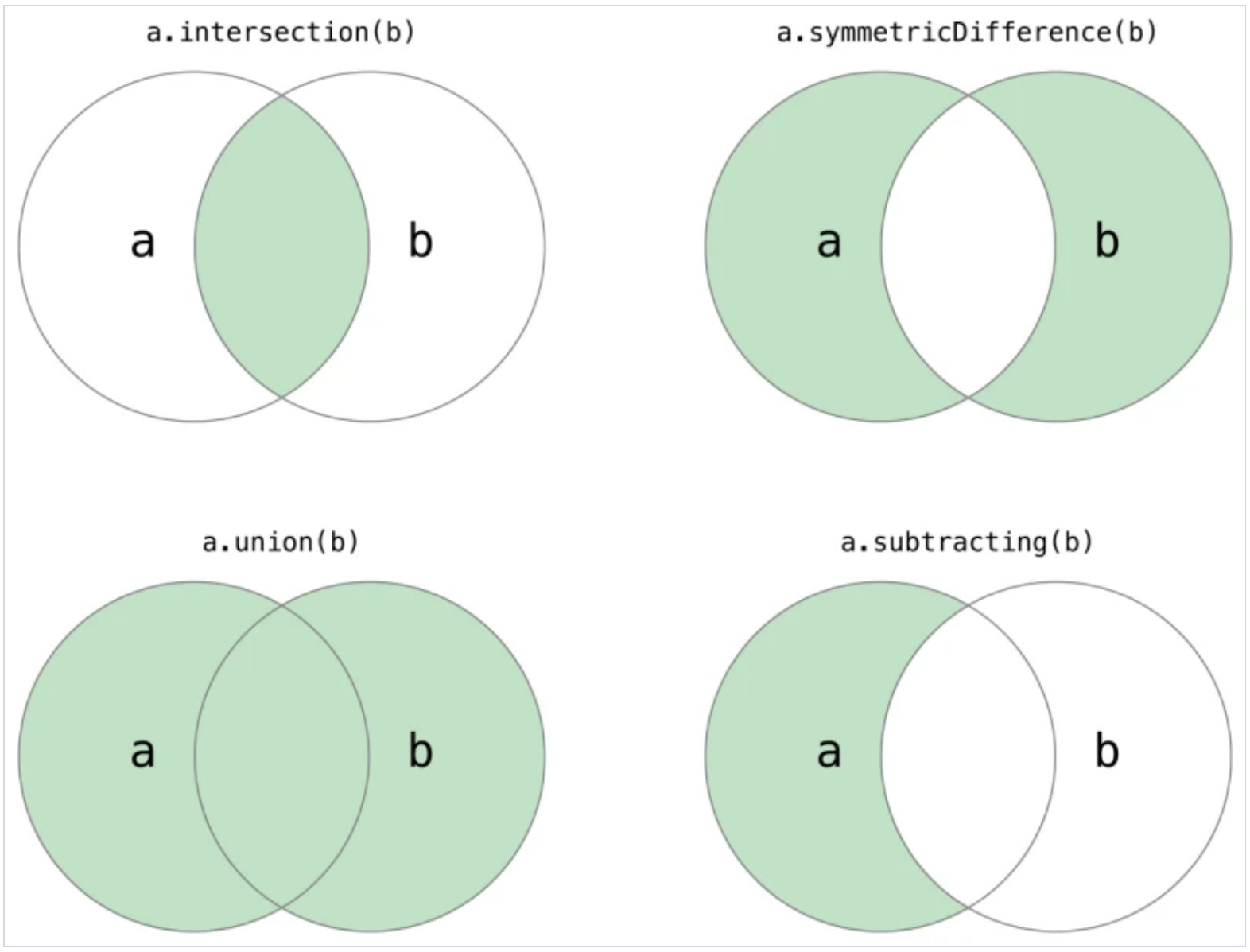
```
User ivan = new User("Ivan", "ivan@email.com");
User petr = new User("Petr", "petr@email.com");

List<User> users = new ArrayList<>();

System.out.println(CollectionUtils.isEmpty(users)); // true
System.out.println(CollectionUtils.isNotEmpty(users)); // false

users.add(ivan);
users.add(petr);

System.out.println(CollectionUtils.isEmpty(users)); // false
System.out.println(CollectionUtils.isNotEmpty(users)); // true
```

```
14
15  users = null;
16  System.out.println(CollectionUtils.isEmpty(users)); // true
17  System.out.println(CollectionUtils.isNotEmpty(users)); // false
```

A Guide to Apache Commons Collections CollectionUtils

Операции над множествами

Если тебе приходилось сталкиваться с теорией множеств, то ты знаешь, что над множествами существуют 4 основные операции: объединение, пересечение, разность и дизъюнкция.



Для этих операций у класса `CollectionUtils` есть 4 метода:

- `union()`
- `intersection()`
- `disjunction()`
- `subtract()`

Ниже будут несколько простых примеров:

Объединение

```
1  List<String> firstList = Arrays.asList("1", "2", "3", "4", "5", "6");
2  List<String> secondList = Arrays.asList("2", "3", "6", "7", "8", "9");
3
4  //объединяем два множества
5  Collection<String> result = CollectionUtils.union(firstList, secondList);
6  System.out.println(ArrayUtils.toString(result));     //[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Пересечение

```
1  List<String> firstList = Arrays.asList("A", "B", "C", "D", "E", "F");
2  List<String> secondList = Arrays.asList("B", "D", "F", "G", "H", "K");
3
4  //пересечения двух множеств
```

```
Collection<String> result = CollectionUtils.intersection(firstList, secondList);
System.out.println(ArrayUtils.toString(result));     //[B, D, F]
```

Дизъюнкция

```
List<String> firstList = Arrays.asList("1", "2", "3", "4", "5", "6");
List<String> secondList = Arrays.asList("2", "3", "6", "7", "8", "9");

//дизъюнкция
Collection<String> result = CollectionUtils.disjunction(firstList, secondList);
System.out.println(ArrayUtils.toString(result)); //[1, 4, 5, 7, 8, 9]
```

Разница (вычет)

```
List<String> firstList = Arrays.asList("1", "2", "3", "4", "5", "6");
List<String> secondList = Arrays.asList("2", "3", "6", "7", "8", "9");

//разница
Collection<String> result = CollectionUtils.subtract(firstList, secondList);
System.out.println(ArrayUtils.toString(result)); // [1, 4, 5]
```

Операции над множествами, продолжение

В этом примере показано, как использовать вышеупомянутые четыре метода для работы с двумя коллекциями:

```
List<Integer> firstList = Arrays.asList(1, 2, 3, 3, 4, 5);
List<Integer> secondList = Arrays.asList(3, 4, 4, 5, 6, 7);

Collection<Integer> union = CollectionUtils.union(firstList, secondList); // Объединение
Collection<Integer> intersection = CollectionUtils.intersection(firstList, secondList); // Пересечение
Collection<Integer> disjunction = CollectionUtils.disjunction(firstList, secondList); // Дизъюнкция
Collection<Integer> subtract = CollectionUtils.subtract(firstList, secondList); // Разница

System.out.println("firstList: " + ArrayUtils.toString(firstList.toArray()));
System.out.println("secondList: " + ArrayUtils.toString(secondList.toArray()));

System.out.println("Объединение: " + ArrayUtils.toString(union.toArray()));
System.out.println("Пересечение: " + ArrayUtils.toString(intersection.toArray()));
System.out.println("Дизъюнкция: " + ArrayUtils.toString(disjunction.toArray()));
System.out.println("Разница: " + ArrayUtils.toString(subtract.toArray()));
```

Наш результат:

```
firstList: {1,2,3,3,4,5}
secondList: {3,4,4,5,6,7}
Объединение: {1,2,3,3,4,4,5,6,7}
Пересечение: {3,4,5}
Дизъюнкция: {1,2,3,4,6,7}
Разница: {1,2,3}
```

Можешь самостоятельно поэкспериментировать с этими методами.

Метод unmodifiableCollection()

Иногда нам приходится реализовывать методы интерфейса, которые должны возвращать коллекции внутренних данных наших объектов. А так как данные внутренние, мы не хотим, чтобы кто-то где-то их менял.

Или мы получили где-то коллекцию, которую не нужно менять, но вынуждены передать ее в какой-то сторонний метод. Где гарантия, что он ее не поменяет?

Для того, чтобы спать спокойно, коллекцию можно обернуть в специальный wrapper, который будет кидать исключение при попытке модифицировать коллекцию.

Пример:

```
1  Collection<String> firstCollection = new ArrayList<String>();
2  Collection<String> secondCollection = Collections.unmodifiableCollection(firstCollection);
3  firstCollection.add("hello");
4  firstCollection.add("from");
5  firstCollection.add("javaRush");
6
7  //secondCollection.add("have a error");
8  System.out.println(secondCollection); //[hello, from, javaRush]
```

Метод `Collections.unmodifiableCollection()` возвращает обертку над переданной ему коллекцией. Если вызывать у нее методы `get()`, `size()`, то все будет работать. Однако при вызове методов `add()`, `set()`, `remove()` ты получишь исключение.

```
java.lang.UnsupportedOperationException
at org.apache.commons.collections.collection.UnmodifiableCollection.add(UnmodifiableCollection.java:75)
```

На самом деле уже сейчас метод помечен как deprecated и вместо него рекомендуется использовать `Collections.unmodifiableCollection(Collection<? extends T> c)`.

```
1  Collection<String> firstCollection = new ArrayList<String>();
2  Collection<String> secondCollection =
3  Collections.unmodifiableCollection(firstCollection);
```

‹ Предыдущая лекция

Следующая лекция ›

Никита Уровень 90 EXPERT 22 сентября 2022, 09:47

https://mvnrepository.com/artifact/commons-collections/commons-collections/3.2.1

Ответить +4