

第四章 CopyOnWriteArraySet源码解析

注：在看这篇文章之前，如果对CopyOnWriteArrayList底层不清楚的话，建议先去看看CopyOnWriteArrayList源码解析。

1、对于CopyOnWriteArraySet需要掌握以下几点

- 创建：CopyOnWriteArraySet()
- 添加元素：即add(E)方法
- 删除对象：即remove(E)方法
- 遍历所有对象：即iterator()，在实际中更常用的是增强型的for循环去做遍历

注：

- CopyOnWriteArraySet（不可添加重复元素）底层是CopyOnWriteArrayList（可添加重复元素）。
- Set集合没有按索引直接获取或修改或添加或删除的方法（eg.get(int index),add(int index,E e),set(int index,E e),remove(int index)）

2、创建

public CopyOnWriteArraySet()

使用方法：

```
Set<String> strSet = new CopyOnWriteArraySet<String>();
```

源代码：

```
private final CopyOnWriteArrayList<E> al; //底层数据结构

public CopyOnWriteArraySet() {
    al = new CopyOnWriteArrayList<E>();
}
```

注意点：

- CopyOnWriteArraySet底层就是一个CopyOnWriteArrayList

3、添加元素

public boolean add(E e)

使用方法：

```
strSet.add("hello")
```


源代码：

```
/**
 * 循环遍历旧数组，若有与e相同的值，return false
 * 若没有，向最后插值
 */
public boolean add(E e) {
    return al.addIfAbsent(e);
}
```

CopyOnWriteArrayList的addIfAbsent(E e)

```
public boolean addIfAbsent(E e) {
    final ReentrantLock lock = this.lock;
    lock.lock();
    try {
        Object[] elements = getArray();
        int len = elements.length;
        Object[] newElements = new Object[len + 1];
        for (int i = 0; i < len; ++i) {
            if (eq(e, elements[i])) //先循环一遍看看有没有与要插入的值相同的值
                return false; // 如果有，直接返回
            else
                newElements[i] = elements[i];
        }
        newElements[len] = e; //如果没有，就赋值
        setArray(newElements);
        return true;
    } finally {
```

```
        lock.unlock();
    }
}
```



注：这一块儿的源代码很简单，只要你看了CopyOnWriteArrayList源码解析中的add方法就能看懂

注意点：

- CopyOnWriteArraySet每次add都要遍历数组，性能要低于CopyOnWriteArrayList


4、删除元素

public boolean remove(Object o)

使用方法：

```
strSet.remove("hello")
```

源代码：




```
/**
 * 调用CopyOnWriteArrayList的remove(Object o)方法
 */
public boolean remove(Object o) {
    return al.remove(o);
}
```

5、遍历所有元素

public Iterator<E> iterator()

使用方法：见上一章《CopyOnWriteArrayList源码解析》

源代码：



```
/**
 * 调用CopyOnWriteArrayList的iterator()
 */
public Iterator<E> iterator() {
    return al.iterator();
}
```

剩余的源代码见上一章《CopyOnWriteArrayList源码解析》

总结：

- CopyOnWriteArraySet底层就是一个CopyOnWriteArrayList
- CopyOnWriteArraySet在add元素的时候要遍历一遍数组，从而起到不添加重复元素的作用，但是由于要遍历数组，效率也会低于CopyOnWriteArrayList的add
- Set集合没有按索引直接获取或修改或添加或删除的方法（eg.get(int index),add(int index,E e),set(int index,E e),remove(int index)）