

Java 深入 ArrayList.toArray(T[] a)

作者: [guihuo](#)

原文链接: <https://ld246.com/article/1544150965990>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

`ArrayList.toArray(T[] a)` 是一个将`ArrayList`转为数组的方法，经常用于方法的参数类型适配。

Java Doc

 返回包含此列表中所有元素的数组；返回数组的运行时类型是指定数组的运行时类型。如果指定的数组能容纳列表，则将该列表返回入参时数组。否则，将分配一个具有指定数组运行时类型和此列表大小的新数组。如果指定的数组能容纳队列，并有剩余的空间（即数组的元素比列表多），那么会将数组中紧接 collection 尾部的元素设置为null。（仅在调用者知道列表中不包含任何 null 元素时才能用此方法确定列表长度）。

源码

```
public <T> T[] toArray(T[] a) {
    if (a.length < size)
        // Make a new array of a's runtimetype, but my contents:
        return (T[]) Arrays.copyOf(elementData, size, a.getClass());
    System.arraycopy(elementData, 0, a, 0, size);
    if (a.length > size)
        a[size] = null;
    return a;
}
```

性能测试

```
public void toArrayPerformanceTest() {
    final int COUNT = 100 * 100 * 100;

    List<Double> list = new ArrayList<>(COUNT);

    for (int i = 0; i < COUNT; i++) {
        list.add(i * 1.0);
    }
    long start = System.nanoTime();

    Double[] notEnoughArray = new Double[COUNT - 1];
    list.toArray(notEnoughArray);

    long middle1 = System.nanoTime();

    Double[] equalArray = new Double[COUNT];
    list.toArray(equalArray);

    long middle2 = System.nanoTime();

    Double[] doubleArray = new Double[COUNT * 2];
    list.toArray(doubleArray);

    long middle3 = System.nanoTime();

    Double[] emptyArray = new Double[0];
    list.toArray(emptyArray);
```

```
long end = System.nanoTime();

double notEnoughArrayTime = middle1 - start;
double equalArrayTime = middle2 - middle1;
double doubleArrayTime = middle3 - middle2;
double emptyArrayTime = end - middle3;

System.out.println("数组容量小于集合大小:notEnoughArrayTime = " + notEnoughArrayTi
e / (1000 * 1000) + " ms");
System.out.println("数组容量等于集合大小:equalArrayTime = " + equalArrayTime / (1000 *
1000) + " ms");
System.out.println("数组容量是集合大小的两倍:doubleArrayTime = " + doubleArrayTime /
1000 * 1000) + " ms");
System.out.println("数组容量是0:emptyArrayTime = " + emptyArrayTime / (1000 * 1000)
" ms");
}
```

总结

测下来数组容量等于集合大小或者为0的时候性能差不多,为最高效的。因为层执行的都是 `System.arraycopy` 甚至入参都差不多。

推荐入参的数组容量等于集合大小,这样不会产生新的数组实例, 同时性能佳。