链滴

Java 中 ArrayList 类的用法

作者：maixiaojie
原文链接：https://ld246.com/article/1559037256208
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)

<p><img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile.com/bing/20180128.jpg?imageView2/1/w/768/h/432/interlace/1/q/100"></p>
<p>1、什么是 ArrayList
ArrayList 就是传说中的动态数组，用 MSDN 中的说法，就是 Array 的复杂版本，它提供了如下一些处：
动态的增加和减少元素
实现了 ICollection 和 IList 接口
灵活的设置数组的大小</p>
<p>2、如何使用 ArrayList
最简单的例子：
ArrayList List = new ArrayList();
for(int i=0;i <10;i++) //给数组增加 10 个 Int 元素
List.Add(i);
//..程序做一些处理
List.RemoveAt(5);//将第 6 个元素移除
for(int i=0;i <3;i++) //再增加 3 个元素
List.Add(i+20);
Int32[] values = (Int32[])List.ToArray(typeof(Int32));//返回 ArrayList 包含的数组</p>
<p>这是一个简单的例子，虽然没有包含 ArrayList 所有的方法，但是可以反映出 ArrayList 最常用用法</p>
<p>3、ArrayList 重要的方法和属性
1）构造器
ArrayList 提供了三个构造器：
public ArrayList();
默认的构造器，将会以默认（16）的大小来初始化内部的数组
public ArrayList(ICollection);
用一个 ICollection 对象来构造，并将该集合的元素添加到 ArrayList
public ArrayList(int);
用指定的大小来初始化内部的数组</p>
<p>2）IsSynchronized 属性和 ArrayList.Synchronized 方法
IsSynchronized 属性指示当前的 ArrayList 实例是否支持线程同步，而 ArrayList.Synchronized 静方法则会返回一个 ArrayList 的线程同步的封装。
如果使用非线程同步的实例，那么在多线程访问的时候，需要自己手动调用 lock 来保持线程同步，如：
ArrayList list = new ArrayList();
//...
lock(list.SyncRoot) //当 ArrayList 为非线程包装的时候，SyncRoot 属性其实就是它自己，但是为满足 ICollection 的 SyncRoot 定义，这里还是使用 SyncRoot 来保持源代码的规范性
{
list.Add(“Add a Item”);
}</p>
<p>如果使用 ArrayList.Synchronized 方法返回的实例，那么就不用考虑线程同步的问题，这个实本身就是线程安全的，实际上 ArrayList 内部实现了一个保证线程同步的内部类，ArrayList.Synchronized 返回的就是这个类的实例，它里面的每个属性都是用了 lock 关键字来保证线程同步。</p>
<p>3）Count 属性和 Capacity 属性
Count 属性是目前 ArrayList 包含的元素的数量，这个属性是只读的。
Capacity 属性是目前 ArrayList 能够包含的最大数量，可以手动的设置这个属性，但是当设置为小于ount 值的时候会引发一个异常。</p>
<p>4）Add、AddRange、Remove、RemoveAt、RemoveRange、Insert、InsertRange br>
这几个方法比较类似
Add 方法用于添加一个元素到当前列表的末尾
AddRange 方法用于添加一批元素到当前列表的末尾
Remove 方法用于删除一个元素，通过元素本身的引用来删除

RemoveAt 方法用于删除一个元素，通过索引值来删除
RemoveRange 用于删除一批元素，通过指定开始的索引和删除的数量来删除
Insert 用于添加一个元素到指定位置，列表后面的元素依次往后移动
InsertRange 用于从指定位置开始添加一批元素，列表后面的元素依次往后移动</p>
<p>另外，还有几个类似的方法：
Clear 方法用于清除现有所有的元素
Contains 方法用来查找某个对象在不在列表之中</p>
<p>其他的我就不一一累赘了，大家可以查看 MSDN，上面讲的更仔细
5）TrimSize 方法
这个方法用于将 ArrayList 固定到实际元素的大小，当动态数组元素确定不在添加的时候，可以调用
个方法来释放空余的内存。
6）ToArray 方法
这个方法把 ArrayList 的元素 Copy 到一个新的数组中。
4、ArrayList 与数组转换
例 1：
ArrayList List = new ArrayList();
List.Add(1);
List.Add(2);
List.Add(3);</p>
<p>Int32[] values = (Int32[])List.ToArray(typeof(Int32));</p>
<p>例 2：
ArrayList List = new ArrayList();
List.Add(1);
List.Add(2);
List.Add(3);</p>
<p>Int32[] values = new Int32[List.Count];
List.CopyTo(values);</p>
<p>上面介绍了两种从 ArrayList 转换到数组的方法</p>
<p>例 3：
ArrayList List = new ArrayList();
List.Add(“string”);
List.Add(1);
//往数组中添加不同类型的元素</p>
<p>object[] values = List.ToArray(typeof(object)); //正确
string[] values = (string[])List.ToArray(typeof(string)); //错误</p>
<p>和数组不一样，因为可以转换为 Object 数组，所以往 ArrayList 里面添加不同类型的元素是不
出错的，但是当调用 ArrayList 方法的时候，要么传递所有元素都可以正确转型的类型或者 Object
型，否则将会抛出无法转型的异常。</p>
<p>5、ArrayList 最佳使用建议
这一节我们来讨论 ArrayList 与数组的差别，以及 ArrayList 的效率问题
1）ArrayList 是 Array 的复杂版本
ArrayList 内部封装了一个 Object 类型的数组，从一般的意义来说，它和数组没有本质的差别，甚</
>
<p>定到实际元素的大小，当动态数组元素确定不在添加的时候，可以调用这个方法来释放空余的内
。
6）ToArray 方法
这个方法把 ArrayList 的元素 Copy 到一个新的数组中。
4、ArrayList 与数组转换
例 1：
ArrayList List = new ArrayList();
List.Add(1);
List.Add(2);
List.Add(3);</p>
<p>Int32[] values = (Int32[])List.ToArray(typeof(Int32));</p>

原文链接：Java 中 ArrayList 类的用法

<p>例 2：
ArrayList List = new ArrayList();
List.Add(1);
List.Add(2);
List.Add(3);</p>
<p>Int32[] values = new Int32[List.Count];
List.CopyTo(values);</p>
<p>上面介绍了两种从 ArrayList 转换到数组的方法</p>
<p>例 3：
ArrayList List = new ArrayList();
List.Add(“string”);
List.Add(1);
//往数组中添加不同类型的元素</p>
<p>object[] values = List.ToArray(typeof(object)); //正确
string[] values = (string[])List.ToArray(typeof(string)); //错误</p>
<p>和数组不一样，因为可以转换为 Object 数组，所以往 ArrayList 里面添加不同类型的元素是不出错的，但是当调用 ArrayList 方法的时候，要么传递所有元素都可以正确转型的类型或者 Object 型，否则将会抛出无法转型的异常。</p>
<p>5、ArrayList 最佳使用建议
这一节我们来讨论 ArrayList 与数组的差别，以及 ArrayList 的效率问题
1）ArrayList 是 Array 的复杂版本
ArrayList 内部封装了一个 Object 类型的数组，从一般的意义来说，它和数组没有本质的差别，甚至 ArrayList 的许多方法，如 Index、IndexOf、Contains、Sort 等都是在内部数组的基础上直接调用 Aray 的对应方法。
2）内部的 Object 类型的影响
对于一般的引用类型来说，这部分的影响不是很大，但是对于值类型来说，往 ArrayList 里面添加和改元素，都会引起装箱和拆箱的操作，频繁的操作可能会影响一部分效率。
但是恰恰对于大多数人，多数的应用都是使用值类型的数组。
消除这个影响是没有办法的，除非你不用它，否则就要承担一部分的效率损失，不过这部分的损失不很大。
3）数组扩容
这是对 ArrayList 效率影响比较大的一个因素。
每当执行 Add、AddRange、Insert、InsertRange 等添加元素的方法，都会检查内部数组的容量是不够了，如果是，它就会以当前容量的两倍来重新构建一个数组，将旧元素 Copy 到新数组中，然后弃旧数组，在这个临界点的扩容操作，应该来说是比较影响效率的。
例 1：比如，一个可能有 200 个元素的数据动态添加到一个以默认 16 个元素大小创建的 ArrayList，将会经过：
162222 = 256
四次的扩容才会满足最终的要求，那么如果一开始就以：
ArrayList List = new ArrayList(210);
的方式创建 ArrayList，不仅会减少 4 次数组创建和 Copy 的操作，还会减少内存使用。</p>
<p>例 2：预计有 30 个元素而创建了一个 ArrayList：
ArrayList List = new ArrayList(30);
在执行过程中，加入了 31 个元素，那么数组会扩充到 60 个元素的大小，而这时候不会有新的元素增加进来，而且有没有调用 TrimSize 方法，那么就有 1 次扩容的操作，并且浪费了 29 个元素大小空间。如果这时候，用：
ArrayList List = new ArrayList(40);
那么一切都解决了。
所以说，正确的预估可能的元素，并且在适当的时候调用 TrimSize 方法是提高 ArrayList 使用效率重要途径。
4）频繁的调用 IndexOf、Contains 等方法（Sort、BinarySearch 等方</p>
<p>法经过优化，不在此列）引起的效率损失
首先，我们要明确一点，ArrayList 是动态数组，它不包括通过 Key 或者 Value 快速访问的算法，所实际上调用 IndexOf、Contains 等方法是执行的简单的循环来查找元素，所以频繁的调用此类方法

不比你自己写循环并且稍作优化来的快，如果有这方面的要求，建议使用 Hashtable 或 SortedList 键值对的集合。
ArrayList al=new ArrayList();</p>
<p>al.Add("How");
al.Add("are");
al.Add("you!");</p>
<p>al.Add(100);
al.Add(200);
al.Add(300);</p>
<p>al.Add(1.2);
al.Add(22.8);</p>

原文链接：Java 中 ArrayList 类的用法