[**java 常用集合list与Set、Map区别及适用场景总结**](http://blog.csdn.net/qq_22118507/article/details/51576319)

标签： [java](http://www.csdn.net/tag/java)[集合](http://www.csdn.net/tag/%e9%9b%86%e5%90%88)[list](http://www.csdn.net/tag/list)[set](http://www.csdn.net/tag/set)[map](http://www.csdn.net/tag/map)

2016-06-03 09:51 32838人阅读 [评论](http://blog.csdn.net/qq_22118507/article/details/51576319#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/qq_22118507/article/details/51576319#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[+]](http://blog.csdn.net/qq_22118507/article/details/51576319)

 转载请备注出自于：http://blog.csdn.net/qq\_22118507/article/details/51576319

**list与Set、Map区别及适用场景**

1、List,Set都是继承自Collection接口，Map则不是

2、List特点：元素有放入顺序，元素可重复 ，Set特点：元素无放入顺序，元素不可重复，重复元素会覆盖掉，（注意：元素虽然无放入顺序，但是元素在set中的位置是有该元素的HashCode决定的，其位置其实是固定的，加入Set 的Object必须定义equals()方法 ，另外list支持for循环，也就是通过下标来遍历，也可以用迭代器，但是set只能用迭代，因为他无序，无法用下标来取得想要的值。）

3.Set和List对比：   
Set：检索元素效率低下，删除和插入效率高，插入和删除不会引起元素位置改变。   
List：和数组类似，List可以动态增长，查找元素效率高，插入删除元素效率低，因为会引起其他元素位置改变。 

4.Map适合储存键值对的数据

5.线程安全集合类与非线程安全集合类

LinkedList、ArrayList、HashSet是非线程安全的，Vector是线程安全的;  
HashMap是非线程安全的，HashTable是线程安全的;  
StringBuilder是非线程安全的，StringBuffer是线程安全的。

下面是具体的使用介绍：

**ArrayList与LinkedList的区别和适用场景**

Arraylist：

优点：ArrayList是实现了基于动态数组的数据结构,因为地址连续，一旦数据存储好了，查询操作效率会比较高（在内存里是连着放的）。

缺点：因为地址连续， ArrayList要移动数据,所以插入和删除操作效率比较低。

LinkedList：

优点：LinkedList基于链表的数据结构,地址是任意的，所以在开辟内存空间的时候不需要等一个连续的地址，对于新增和删除操作add和remove，LinedList比较占优势。LinkedList 适用于要头尾操作或插入指定位置的场景

缺点：因为LinkedList要移动指针,所以查询操作性能比较低。

适用场景分析：

 当需要对数据进行对此访问的情况下选用ArrayList，当需要对数据进行多次增加删除修改时采用LinkedList。

**ArrayList与Vector的区别和适用场景**

 ArrayList有三个构造方法：

**Java代码**

1. **public** ArrayList(**int** initialCapacity)//构造一个具有指定初始容量的空列表。
2. **public** ArrayList()//构造一个初始容量为10的空列表。
3. **public** ArrayList(Collection<? **extends** E> c)//构造一个包含指定 collection 的元素的列表

 Vector有四个构造方法：

**Java代码**

1. **public** Vector()//使用指定的初始容量和等于零的容量增量构造一个空向量。
2. **public** Vector(**int** initialCapacity)//构造一个空向量，使其内部数据数组的大小，其标准容量增量为零。
3. **public** Vector(Collection<? **extends** E> c)//构造一个包含指定 collection 中的元素的向量
4. **public** Vector(**int** initialCapacity,**int** capacityIncrement)//使用指定的初始容量和容量增量构造一个空的向量

ArrayList和Vector都是用数组实现的，主要有这么三个区别：

1.Vector是多线程安全的，线程安全就是说多线程访问同一代码，不会产生不确定的结果。而ArrayList不是，这个可以从源码中看出，Vector类中的方法很多有synchronized进行修饰，这样就导致了Vector在效率上无法与ArrayList相比；

2.两个都是采用的线性连续空间存储元素，但是当空间不足的时候，两个类的增加方式是不同。

3.Vector可以设置增长因子，而ArrayList不可以。

4.Vector是一种老的动态数组，是线程同步的，效率很低，一般不赞成使用。

适用场景分析：

1.Vector是线程同步的，所以它也是线程安全的，而ArrayList是线程异步的，是不安全的。如果不考虑到线程的安全因素，一般用ArrayList效率比较高。  
2.如果集合中的元素的数目大于目前集合数组的长度时，在集合中使用数据量比较大的数据，用Vector有一定的优势。

**HashSet与Treeset的适用场景**

1.TreeSet 是二差树（红黑树的树据结构）实现的,Treeset中的数据是自动排好序的，不允许放入null值   
  
2.HashSet 是哈希表实现的,HashSet中的数据是无序的，可以放入null，但只能放入一个null，两者中的值都不能重复，就如数据库中唯一约束

3.HashSet要求放入的对象必须实现HashCode()方法，放入的对象，是以hashcode码作为标识的，而具有相同内容的String对象，hashcode是一样，所以放入的内容不能重复。但是同一个类的对象可以放入不同的实例

   适用场景分析:HashSet是基于Hash算法实现的，其性能通常都优于TreeSet。为快速查找而设计的Set，我们通常都应该使用HashSet，在我们需要排序的功能时，我们才使用TreeSet。

**HashMap与TreeMap、HashTable的区别及适用场景**

HashMap 非线程安全

HashMap：基于哈希表实现。使用HashMap要求添加的键类明确定义了hashCode()和equals()[可以重写hashCode()和equals()]，为了优化HashMap空间的使用，您可以调优初始容量和负载因子。

TreeMap：非线程安全基于红黑树实现。TreeMap没有调优选项，因为该树总处于平衡状态。

适用场景分析：

HashMap和HashTable:HashMap去掉了HashTable的contains方法，但是加上了containsValue()和containsKey()方法。HashTable同步的，而HashMap是非同步的，效率上比HashTable要高。HashMap允许空键值，而HashTable不允许。

HashMap：适用于Map中插入、删除和定位元素。

Treemap：适用于按自然顺序或自定义顺序遍历键(key)。