HashMap和Hashtable的比较是Java面试中的常见问题，用来考验程序员是否能够正确使用集合类以及是否可以随机应变使用多种思路解决问题。HashMap的工作原理、ArrayList与Vector的比较以及这个问题是有关Java 集合框架的最经典的问题。Hashtable是个过时的集合类，存在于Java API中很久了。在Java 4中被重写了，实现了Map接口，所以自此以后也成了Java集合框架中的一部分。Hashtable和HashMap在Java面试中相当容易被问到，甚至成为了集合框架面试题中最常被考的问题，所以在参加任何Java面试之前，都不要忘了准备这一题。

这篇文章中，我们不仅将会看到HashMap和Hashtable的区别，还将看到它们之间的相似之处。

### HashMap和Hashtable的区别

HashMap和Hashtable都实现了Map接口，但决定用哪一个之前先要弄清楚它们之间的分别。主要的区别有：线程安全性，同步(synchronization)，以及速度。

1. HashMap几乎可以等价于Hashtable，除了HashMap是非synchronized的，并可以接受null(HashMap可以接受为null的键值(key)和值(value)，而Hashtable则不行)。
2. HashMap是非synchronized，而Hashtable是synchronized，这意味着Hashtable是线程安全的，多个线程可以共享一个Hashtable；而如果没有正确的同步的话，多个线程是不能共享HashMap的。Java 5提供了ConcurrentHashMap，它是HashTable的替代，比HashTable的扩展性更好。
3. 另一个区别是HashMap的迭代器(Iterator)是fail-fast迭代器，而Hashtable的enumerator迭代器不是fail-fast的。所以当有其它线程改变了HashMap的结构（增加或者移除元素），将会抛出ConcurrentModificationException，但迭代器本身的remove()方法移除元素则不会抛出ConcurrentModificationException异常。但这并不是一个一定发生的行为，要看JVM。这条同样也是Enumeration和Iterator的区别。
4. 由于Hashtable是线程安全的也是synchronized，所以在单线程环境下它比HashMap要慢。如果你不需要同步，只需要单一线程，那么使用HashMap性能要好过Hashtable。
5. HashMap不能保证随着时间的推移Map中的元素次序是不变的。

### 要注意的一些重要术语：

1) sychronized意味着在一次仅有一个线程能够更改Hashtable。就是说任何线程要更新Hashtable时要首先获得同步锁，其它线程要等到同步锁被释放之后才能再次获得同步锁更新Hashtable。

2) Fail-safe和iterator迭代器相关。如果某个集合对象创建了Iterator或者ListIterator，然后其它的线程试图“结构上”更改集合对象，将会抛出ConcurrentModificationException异常。但其它线程可以通过set()方法更改集合对象是允许的，因为这并没有从“结构上”更改集合。但是假如已经从结构上进行了更改，再调用set()方法，将会抛出IllegalArgumentException异常。

3) 结构上的更改指的是删除或者插入一个元素，这样会影响到map的结构。

### 我们能否让HashMap同步？

HashMap可以通过下面的语句进行同步：  
Map m = Collections.synchronizeMap(hashMap);

### 结论

Hashtable和HashMap有几个主要的不同：线程安全以及速度。仅在你需要完全的线程安全的时候使用Hashtable，而如果你使用Java 5或以上的话，请使用ConcurrentHashMap吧。

原文链接： [Javarevisited](http://javarevisited.blogspot.hk/2010/10/difference-between-hashmap-and.html" \t "http://www.importnew.com/_blank) 翻译： [ImportNew.com](http://www.importnew.com/)- [唐小娟](http://www.importnew.com/author/tangxiaojuan)  
译文链接： <http://www.importnew.com/7010.html>  
[ **转载请保留原文出处、译者和译文链接。**]