Guava学习之Immutable集合

Immutable中文意思就是不可变。那为什么需要构建一个不可变的对象？原因有以下几点：

1. 在并发程序中，使用Immutable既保证线程安全性，也大大增强了并发时的效率（跟并发锁方式相比）。尤其当一个对象是值对象时，更应该考虑采用Immutable方式；
2. 被不可信类库使用时会很安全；
3. 如果一个对象不需要支持修改操作(mutation)，将会节省空间和时间的开销；经过分析，所有不可变的集合实现都比可变集合更加有效地利用内存；
4. 可以当作一个常量来对待，并且这个对象在以后也不会被改变。

将一个对象复制一份成immutable的，是一个防御性编程技术。在JDK类库中很多集合(List、Set、Map等)都可以调用Collections类提供的静态方法unmodifiableXXX(...)来得到一个不可修改的视图，如下所示：

```
List list = new ArrayList();  
list.add("wyp");  
list.add("good");
```

```
List unmodifiableList = Collections.unmodifiableList(list);  
System.out.println(unmodifiableList);//[wyp, good]  
unmodifiableList.add("add");
```

上述代码利用Collections.unmodifiableList(list)得到一个不可修改的集合unmodifiableList，当unmodifiableList.add("add")时，运行代码将会出现以下异常：

```
Exception in thread "main" java.lang.UnsupportedOperationException  
at java.util.Collections$UnmodifiableCollection.add(Collections.java:1018)  
at com.wyp.test.testFiles(test.java:152)  
at com.wyp.test.main(test.java:160)
```

一切看起来很不错，因为调用unmodifiableList.add()会抛出一个java.lang.UnsupportedOperationException。但，如果有用户修改了list，会发生什么情况？在上述代码的下面加入以下代码：

```
list.add("add");  
System.out.println(unmodifiableList);
```

当你再次打印unmodifiableList的时候，你会发现结果是[wyp, good, add]，多了一个add元素。unmodifiableList不是不可变的吗？这显然不是我们期望的。

说明

: Collections.unmodifiableList(...)实现的不是真正的不可变集合，当原始集合被修改后，不可变集合里面的元素也是跟着发生变化。

利用JDK类库中提供的unmodifiableXXX方法最少存在以下几点不足：

1. 笨拙：因为你每次都得写那么多代码；
2. 不安全：如果没有引用到原来的集合，这种情况之下才会返回唯一真正永恒不变的集合；
3. 效率很低：返回的不可修改的集合数据结构仍然具有可变集合的所有开销。

Guava类库中提供的Immutable才是真正的不可修改的集合。

```
import com.google.common.collect.ImmutableList;
```

```
ImmutableList immutableList = ImmutableList.of("wyp", "good");
```

当你往immutableList 中添加元素，也会抛出java.lang.UnsupportedOperationException异常。

可以用ImmutableList提供的以下几个方法来得到一个不可变的集合：

copyOf(Collection<? extends E> elements)比如ImmutableList.copyOf(list);

用of()方法得到一个空的不可变的List(ImmutableList提供了很多of()的重写)；

利用Builder来实现：

```
List list = new ArrayList();  
list.add("wyp");  
list.add("good");  
ImmutableList GOOGLE_IMMUTABLE = ImmutableList  
    . builder().addAll(list).add("add").build();
```

JDK和Guava提供的不可变类型：

可变集合类型

Collection

List

Set

JDK or Guava?

JDK

JDK

JDK

Guava不可变集合

ImmutableCollection

ImmutableList

ImmutableSet

SortedSet/NavigableSet	JDK	ImmutableSortedSet
Map	JDK	ImmutableMap
SortedMap	JDK	ImmutableSortedMap
Multiset	Guava	ImmutableMultiset
SortedMultiset	Guava	ImmutableSortedMultiset
Multimap	Guava	ImmutableMultimap
ListMultimap	Guava	ImmutableListMultimap
SetMultimap	Guava	ImmutableSetMultimap
BiMap	Guava	ImmutableBiMap
ClassToInstanceMap	Guava	ImmutableClassToInstanceMap
Table	Guava	ImmutableTable

本博客文章除特别声明，全部都是原创！

原创文章版权归过往记忆大数据（[过往记忆](#)）所有，未经许可不得转载。

本文链接: [【】](#)（）