

## Guava学习之BiMap

在前面的《[Guava学习之Multimap](#)》文章中我们谈到了Guava类库中的Multimap，其特点是存在在Multimap中的键值对可以不唯一；而我们又知道，在Java集合类库中有个Map，它的特点是存放的键（Key）是唯一的，而值（Value）可以不唯一，如果我们需要键（Key）和值（Value）都唯一，该怎么实现？这就是今天要谈的BiMap结构。

在过去，如果需要将Map结构中的键值对反转（也就是key->value转变成value->key），这时候我们需要定义两个Map数据结构来存储。但，如果Map中存在多个value相同的元素会发生什么情况呢？这时候添加进去的key将会覆盖先前加进去的key。如下所示：

k1->v1, k2->v2, k3->v3, k4->v1如果我们需反转key和Value将会变成v1->k4, v2->k2, v3->k3, 细心的人会发现，怎么少了一项，这是为什么呢？因为k1->v1和k4->v1反转之后变成v1->k1和v1->k4，但由于Map的特点v1->k4将覆盖先前的v1->k1。

这就是BiMap用武之地了！BiMap<K, V>是Map<K, V>类型的数据类型（因为BiMap实现了java.util.Map接口，注意和Multimap的区别，Multimap没实现Map的接口）。它的特点是它的value和它key一样也是不可重复的，换句话说它的key和value是等价的。如果你往BiMap的value里面放了重复的元素，就会得到IllegalArgumentException。如下：

```
BiMap<String, String> upperToSmall = HashBiMap.create();
upperToSmall.put("A", "a");
upperToSmall.put("B", "b");
upperToSmall.put("C", "c");
System.out.println(upperToSmall.get("A"));
```

将得到一个存放key和value唯一的结构，如果在上述代码加入一行

```
upperToSmall.put("D", "c");
```

则程序将抛出以下异常

```
Exception in thread "main" java.lang.IllegalArgumentException: value already present: c
at com.google.common.collect.HashBiMap.put(HashBiMap.java:241)
at com.google.common.collect.HashBiMap.put(HashBiMap.java:218)
at com.wyp.test.testFiles(test.java:142)
```

at com.wyp.test.main(test.java:153)

这是因为BiMap的强制唯一性：BiMap强制其value的唯一性，如果发现违规则会抛出 `IllegalArgumentException`。是不是当添加的value先前已经有了就不能添加了呢？答案是否，我们可以利用BiMap提供的 `BiMap.forcePut(key, value)` 来实现。那如何通过BiMap将key和value反转呢？很简单，代码如下：

```
BiMap<String, String> smallToUpper = upperToSmall.inverse();  
System.out.println(smallToUpper.get("a")); //
```

这样就实现了键值反转！需要注意的是，`inverse`方法返回一个反转后的BiMap，即key/value互相切换的映射。这个反转的map并不是一个新的map（`upperToSmall == upperToSmall.inverse().inverse()`），而是一个视图，这意味着，你在这个反转后的map中的任何增删改操作都会影响原来的map。

BiMap的常用实现有：

1. `HashBiMap`: key 集合与 value 集合都有 `HashMap` 实现
2. `EnumBiMap`: key 与 value 都必须是 `enum` 类型
3. `ImmutableBiMap`: 不可修改的 `BiMap`

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: [【】（）](#)