Guava学习之Multimap

相信大家对Java中的Map类及其之类有大致的了解，Map类是以键值对的形式来存储元素（Key->Value），但是熟悉Map的人都知道，Map中存储的Key是唯一的。什么意思呢？就是假如我们有两个key相同，但value不同的元素需要插入到map中去，那么先前的key对应的value将会被后来的值替换掉。如果我们需要用Map来把相同key的值存在一起，代码看起来像下面一样：

```
package com.wyp.Map;

/**
 *@User: 过往记忆
 *@Date: 2013-7-9
 * Blog:http://iteblog.com
 * Email: wyphao.2007@163.com
 *
 **/

public class Person {
 //姓名
 private String name;
 //年龄
 private int age;
 //性别
 private String sex;

 public Person() {

  // TODO Auto-generated constructor stub
 }

 /**
  * @param name
  * @param age
  * @param sex
  */
 public Person(String name, int age, String sex) {
  super();
  this.name = name;
  this.age = age;
  this.sex = sex;
 }

 public String getName() {
```

```
 return name;
}

public void setName(String name) {
 this.name = name;
}

public int getAge() {
 return age;
}

public void setAge(int age) {
 this.age = age;
}

public String getSex() {
 return sex;
}

public void setSex(String sex) {
 this.sex = sex;
}

@Override
public String toString() {
 return "Person [name=" + name + ",
                 age=" + age + ", sex=" + sex + "]";
}
}
```

下面进行

```
//性别统计
public void genderStatistics(List personList){
 if(personList == null){
  return;
 }

 Map<String, List> map = new HashMap<String, List>();
 for(Person person : personList){
  String sex = person.getSex();
  List persons = map.get(sex);
  if(persons == null){//第一次加入
   persons = new ArrayList();
  }

  persons.add(person);
```

```
  map.put(sex, persons);
 }

 for (Entry<String, List> entry : map.entrySet()) {
  String key = entry.getKey();
  System.out.println(key + "₩t" + entry.getValue());
 }
}
```

虽然实现了功能，但是代码比较长，但是如果你用Guava去实现同样的功能，你会发现你的代码一下子变少了。Guava提供了下面的结构

```
import com.google.common.collect.ArrayListMultimap;
import com.google.common.collect.Multimap;

Multimap<K, V> myMultimap = ArrayListMultimap.create();
```

从名字可以看出，Multimap可以存放的key值是不唯一的，Multimap并没有实现 Map 的接口，所以不需要达到键唯一的要求。如果存放了key一样的元素，Multimap并不会覆盖以前相同的key元素，而是加进去了，其结果有点像{k1=[v1, v2, v3], k2=[v7, v8],....}其中v1, v2, v3对应的key都是k1（第二种理解的方法是一个key对应一个value[没有说唯一]，所以其存储结果有点像k1 -> v1 k1 -> v2 k1 -> v3 k2 -> v7 k2 -> v8），而如果是Map，则它的结果有点像{k1=v1, k2=v2,...}看到区别了吧？那么，用Multimap实现上面同样的功能代码有点像

```
Multimap<String, Person> myMultimap = ArrayListMultimap.create();
for (Person person : personList) {
 String sex = person.getSex();
 myMultimap.put(sex, person);
}

Map<String, Collection> map1 = myMultimap.asMap();
for (Entry<String, Collection> entry : map1.entrySet()) {
 String key = entry.getKey();
 System.out.println(key + "₩t" + entry.getValue());
}
```

看到了吧，代码简单多了吧！这里有一点你可能会疑惑，就是为何get方法返回的是一个Collectio

n而不是list，这是因为前者会更加有用。如果你需要基于multimap直接操作list或者set，那么可以在定义类型的时候使用子类名称：ListMultimap，SetMultimap和SortedSetMultimap。例如：

```
ListMutlimap<String,Person> myMutlimap = ArrayListMultimap.create();
// Returns a List, not a Collection.
List myValues = myMutlimap.get("myKey");
```

这里需要再次强调的是，Multimap不是Map（Multimap Is Not A Map）！

一个Multimap<K, V>不是一个Map<K, Collection<V>>, 虽然我们可以利用Map<K, Collection<V>>来实现Multimap<K, V>，即使如此，它们之间还是有区别的：

1. Multimap.get(key) 总是返回一个unll值（可能是一个空的collection）；
2. 可以利用asMap()方法来得到一个 Map<K, Collection<V>>类型的数据（或者利用ListMultimap中的静态方法Multimaps.asMap()得到一个Map<K, List<V>类型的数据； SetMultimap和SortedSetMultimap也类似）；
3. Multimap.containsKey(key)只有在这个key和一个或者多个元素相关联的时候才会返回true，如果这个key在删除之前和一个或者多个元素相关联则函数将会返回false；
4. Multimap.entries()返回Multimap所有实体的所有key值；
5. Multimap.size()返回在Multimap中存放的所有实体的数量，而不是不同keys的数量。我们可以利用Multimap.keySet().size()得到Multimap中所有不同keys的数量。

(完)