guava 学习笔记（二） 瓜娃（guava）的API快速熟悉使用

1，大纲

让我们来熟悉瓜娃，并体验下它的一些API,分成如下几个部分：

- **Introduction**
- **Guava Collection API**
- **Guava Basic Utilities**
- **IO API**
- **Cache API**

2，为神马选择瓜娃?

- 瓜娃是java API蛋糕上的冰激凌（精华）
- 高效设计良好的API.
- 被google的开发者设计，实现和使用。
- 遵循高效的java这本书的好的语法实践。
- 使代码更刻度，简洁，简单。
- 使用java 1.5的特性，

- 流行的API，动态的开发
- 它提供了大量相关的应用类，集合，多线程，比较，字符串，输入输出，缓存，网络，原生类型，数学，反射等等
- 百分百的单元测试，被很多的项目使用，帮助开发者专注业务逻辑而不是写java应用类
- 节省时间，资源，提高生产力
- 我的目的是为基本的java特征提供开源代码的支持，而不是自己再写一个
- Apache Common库-Apache是一个很好的成熟的库，但是不支持泛型，Apache对早起的java版本很有用，（1.5之前的）
- java7，java8 最新的java支持一些guava的API

guava最新的正式版本是14.0-rc2，这个版本需要java1.6支持.

最新的maven坐标是：

```
<dependency>
<groupId>com.google.guava</groupId>
<artifactId>guava</artifactId>
<version>14.0-rc2</version>
</dependency>
```

3,集合API的使用

3.1简化工作

可以简化集合的创建和初始化；

类别	原来的写法	guava的写法
集合创建	Map<String, Map<String, String>> map = new HashMap<String, Map<String,String>>(); List<List<Map<String, String>>> list = new	Map<String, Map<String, String>> map = Maps.newHashMap(); List<List<Map<String, String>>> list = Lists.newArrayList(); //1,简化集合的创建 List<Person> personList= Lists.newLinkedList();

	ArrayList<List<Map<String,String>>>(); 	Set<Person> personSet= Sets.newHashSet(); Map<String,Person> personMap= Maps.newHashMap(); Integer[] intArrays= ObjectArrays.newArray(Integer.class,10);

集合初始化	Set<String> set = new HashSet<String>();
	set.add("one");
	set.add("two");
	set.add("three");
		Set<String> set = Sets.newHashSet("one","two","three");
		List<String> list = Lists.newArrayList("one","two","three");
		Map<String, String> map = ImmutableMap.of("ON","TRUE","OFF","FALSE");
		//2,简化集合的初始化
		List<Person> personList2= Lists.newArrayList(new Person(1, 1, "a", "46546", 1, 20),
		new Person(2, 1, "a", "46546", 1, 20));
		Set<Person> personSet2= Sets.newHashSet(new Person(1,1,"a","46546",1,20),
		new Person(2,1,"a","46546",1,20));
		Map<String,Person> personMap2= ImmutableMap.of("hello",new Person(1,1,"a","46546",1,20),"fuck",new Person(2,1,"a","46546",1,20));

3.2 不变性

很大一部分是google集合提供了不变性，不变对比可变：

- 数据不可改变
- 线程安全
- 不需要同步逻辑

Ext
Hibernate/Nhibernate
Java(23)
Javascript(3)
java基础(2)
Linq
MVC
scala(1)
struts2(3)
笔试题上机(1)
编程笔记(10)
反射技术
面试(1)
设计模式
数据结构
数据库(5)

随笔档案(83)

2017年1月 (3)
2016年11月 (1)
2016年4月 (1)
2016年3月 (2)
2015年9月 (2)
2015年3月 (1)
2014年8月 (1)
2014年7月 (1)
2014年6月 (1)
2014年4月 (1)
2014年3月 (10)

- 可以被自由的共享
- 容易设计和实现
- 内存和时间高效

不变对比不可修改

google的不变被确保真正不可改变，而不可修改实际上还是可以修改数据；如下所示：

```
Set<Integer> data = new HashSet<Integer>();
data.addAll(Arrays.asList(10, 20, 30, 40, 50, 60, 70, 80));
Set<Integer> fixedData = Collections.unmodifiableSet(data); // fixedData - [50, 70, 80, 20, 40, 10, 60, 30]
data.add(90); // fixedData - [50, 70, 80, 20, 40, 10, 90, 60, 30]
```

如何创建不可变的集合：

```
ImmutableSet<Integer> numbers = ImmutableSet.of(10, 20, 30, 40, 50);
```

使用copyOf方法

```
ImmutableSet<Integer> another = mmutableSet.copyOf(numbers);
```

使用Builder方法

```
ImmutableSet<Integer> numbers2 = ImmutableSet.<Integer>builder().addAll(numbers) .add(60) .add(70).add(80).build();
```

```
//3,创建不可变的集合
ImmutableList<Person> personImmutableList=
ImmutableList.of(new Person(1, 1, "a", "46546", 1, 20), new Person(2, 1, "a", "46546", 1, 20));
ImmutableSet<Person> personImmutableSet=ImmutableSet.copyOf(personSet2);
ImmutableMap<String,Person> personImmutableMap=ImmutableMap.<String,Person>builder()
.put("hell",new Person(1,1,"a","46546",1,20)).putAll(personMap2) .build();
```

2013年11月 (6)
2013年8月 (1)
2013年6月 (3)
2013年2月 (3)
2013年1月 (1)
2012年10月 (1)
2012年7月 (6)
2012年6月 (1)
2012年5月 (3)
2012年4月 (3)
2012年3月 (1)
2012年2月 (1)
2012年1月 (5)
2011年10月 (6)
2011年9月 (1)
2011年8月 (3)
2011年6月 (1)
2011年3月 (6)
2011年2月 (4)
2011年1月 (3)

文章档案(11)

2013年6月 (1)
2011年3月 (1)
2011年2月 (6)
2011年1月 (3)

相册

一些图片

3.3 新的集合类型

The Guava API provides very useful new collection types that work very nicely with existing java collections.

guava API 提供了有用的新的集合类型，协同已经存在的java集合工作的很好。

分别是 MultiMap， MultiSet， Table， BiMap， ClassToInstanceMap

种类	写的例子

MultiMap	一种key可以重复的map，子类有ListMultimap和SetMultimap，对应的通过key分别得到list和set
	
	Multimap<String, Person> customersByType =ArrayListMultimap.create();customersByType.put("abc", new Person(1, 1, "a", "46546", 1, 20));
	customersByType.put("abc", new Person(1, 1, "a", "46546", 1, 30));
	customersByType.put("abc", new Person(1, 1, "a", "46546", 1, 40));
	customersByType.put("abc", new Person(1, 1, "a", "46546", 1, 50));
	customersByType.put("abcd", new Person(1, 1, "a", "46546", 1, 50));
	customersByType.put("abcde", new Person(1, 1, "a", "46546", 1, 50));
	for(Person person:customersByType.get("abc"))
	{
	System.out.println(person.getAge());
	}

MultiSet	不是集合，可以增加重复的元素，并且可以统计出重复元素的个数，例子如下:
	private static void testMulitiSet() {
	Multiset<Integer> multiSet = HashMultiset.create();
	multiSet.add(10);
	multiSet.add(30);
	multiSet.add(30);
	multiSet.add(40);

积分与排名

积分 - 88845
排名 - 4317

最新评论

1. Re:怎么阅读别人的项目代码?

会看懂app项目吗？帮我看看

--做一个奋斗的小帮主

2. Re:shiro的使用1 简单的认证

你牛，为什么我学shiro 好难啊，不知道哪里入手，概念看了一大堆，demo也跑了，结果懵逼了。。。

--小白种白菜

3. Re:guava 学习笔记（二） 瓜娃（guava）的API快速熟悉使用

原来还有这么好滴的库啊!

--rgqancy

4. Re:跨域的简单研究

Ah。。。不错的呢。我们公司里的普遍做法比较老土的。。。主要写SERVICE层从后段去访问那些跨域的API，然后在同域中返回给自己的应用。。。或者在以特定DNS（Access-Control-Allo.....

--Will.Hu

5. Re:丰巢面试小结

强烈同意：作者所说的引用"这个问题我确实是存在的，不太精通的问题，也聊了起来。但是，开源工具千千万万，不可能每个都专研到源码和设计原理级别的，更多的是基于使用场景。给我一个场景，我通过查文档和资料，熟.....

--Will.Hu

	System.out.println(multiSet.count(30)); // 2
	System.out.println(multiSet.size()); //4
	}

Table	相当于有两个key的map，不多解释
	
	private static void testTable() {
	Table<Integer,Integer,Person> personTable=HashBasedTable.create();
	personTable.put(1,20,new Person(1, 1, "a", "46546", 1, 20));
	personTable.put(0,30,new Person(2, 1, "ab", "46546", 0, 30));
	personTable.put(0,25,new Person(3, 1, "abc", "46546", 0, 25));
	personTable.put(1,50,new Person(4, 1, "aef", "46546", 1, 50));
	personTable.put(0,27,new Person(5, 1, "ade", "46546",0, 27));
	personTable.put(1,29,new Person(6, 1, "acc", "46546", 1, 29));
	personTable.put(0,33,new Person(7, 1, "add", "46546",0, 33));
	personTable.put(1,66,new Person(8, 1, "afadsf", "46546", 1, 66));
	
	//1,得到行集合
	Map<Integer,Person> rowMap= personTable.row(0);
	int maxAge= Collections.max(rowMap.keySet());
	
	}

BiMap	是一个一一映射，可以通过key得到value，也可以通过value得到key；
	
	private static void testBitMap() {
	//双向map
	BiMap<Integer,String> biMap=HashBiMap.create();
	
	biMap.put(1,"hello");
	biMap.put(2,"helloa");
	biMap.put(3,"world");
	biMap.put(4,"worldb");
	biMap.put(5,"my");
	biMap.put(6,"myc");

	int value= biMap.inverse().get("my");
	System.out.println("my --"+value);
	}

ClassToInstanceMap	有的时候，你的map的key并不是一种类型，他们是很多类型，你想通过映射他们得到这种类型，guava提供了ClassToInstanceMap满足了这个目的。
	
	除了继承自Map接口，ClassToInstaceMap提供了方法 T getInstance(Class<T>) 和 T putInstance(Class<T>, T),消除了强制类型转换。
	
	该类有一个简单类型的参数，通常称为B，代表了map控制的上层绑定，例如：
	
	ClassToInstanceMap<Number> numberDefaults = MutableClassToInstanceMap.create();
	numberDefaults.putInstance(Integer.class, Integer.valueOf(0));
	从技术上来说，ClassToInstanceMap<B> 实现了Map<Class<? extends B>, B>，或者说，这是一个从B的子类到B对象的映射，这可能使得ClassToInstanceMap的泛型轻度混乱，但是只要记住B总是Map的上层绑定类型，通常来说B只是一个对象。
	guava提供了有用的实现， MutableClassToInstanceMap 和 ImmutableClassToInstanceMap.
	重点：像其他的Map<Class,Object>,ClassToInstanceMap 含有的原生类型的项目，一个原生类型和他的相应的包装类可以映射到不同的值；
	
	private static void testClass() {
	ClassToInstanceMap<Person> classToInstanceMap =MutableClassToInstanceMap.create();
	
	Person person= new Person(1,20,"abc","46464",1,100);
	
	classToInstanceMap.putInstance(Person.class,person);
	
	// System.out.println("string:"+classToInstanceMap.getInstance(String.class));
	// System.out.println("integer:" + classToInstanceMap.getInstance(Integer.class));
	
	Person person1=classToInstanceMap.getInstance(Person.class);
	
	}

3.4 谓词和筛选

谓词（Predicate）是用来筛选集合的；

谓词是一个简单的接口，只有一个方法返回布尔值，但是他是一个很令人惊讶的集合方法，当你结合collections2.filter方法使用，这个筛选方法返回原来的集合中满足这个谓词接口的元素；

举个例子来说：筛选出集合中的女人

```
public static void main(String[] args)
{
Optional<ImmutableMultiset<Person>> optional=Optional.fromNullable(testPredict());

if(optional.isPresent())
{
for(Person p:optional.get())
{
System.out.println("女人："+p);
}
}

System.out.println(optional.isPresent());
}


public static ImmutableMultiset<Person> testPredict()
{
List<Person> personList=Lists.newArrayList(new Person(1, 1, "a", "46546", 1, 20),
new Person(2, 1, "ab", "46546", 0, 30),
new Person(3, 1, "abc", "46546", 0, 25),
new Person(4, 1, "aef", "46546", 1, 50),
new Person(5, 1, "ade", "46546",0, 27),
new Person(6, 1, "acc", "46546", 1, 29),
new Person(7, 1, "add", "46546",0, 33));
```

```
return ImmutableMultiset.copyOf(Collections2.filter(personList,new Predicate<Person>() {
@Override
public boolean apply( Person input) {
return input.getSex()==0;
}
}));
}
```

Predicates含有一些内置的筛选方法，比如说 in ,and ,not等，根据实际情况选择使用。

3.5 功能和转换

转换一个集合为另外一个集合;

实例如下:

```
public static void main(String[] args)
{
Optional<ImmutableMultiset<String>> optional=Optional.fromNullable(testTransform());

if(optional.isPresent())
{
for(String p:optional.get())
{
System.out.println("名字: "+p);
}
}

System.out.println(optional.isPresent());
}


public static ImmutableMultiset<String> testTransform()
{
List<Person> personList=Lists.newArrayList(new Person(1, 1, "a", "46546", 1, 20),
new Person(2, 1, "ab", "46546", 0, 30),
new Person(3, 1, "abc", "46546", 0, 25),
new Person(4, 1, "aef", "46546", 1, 50),
```

```
new Person(5, 1, "ade", "46546",0, 27),
new Person(6, 1, "acc", "46546", 1, 29),
new Person(7, 1, "add", "46546",0, 33));

return ImmutableMultiset.copyOf(Lists.transform(personList,new Function<Person, String>() {
@Override
public String apply( Person input) {
return input.getName();
}
}));
}
```

3.6 排序

是guava一份非常灵活的比较类，可以被用来操作，扩展，当作比较器，排序提供了集合排序的很多控制；

实例如下：

```
Lists.newArrayList(30, 20, 60, 80, 10);

Ordering.natural().sortedCopy(numbers); //10,20,30,60,80

Ordering.natural().reverse().sortedCopy(numbers); //80,60,30,20,10

Ordering.natural().min(numbers); //10

Ordering.natural().max(numbers); //80

Lists.newArrayList(30, 20, 60, 80, null, 10);

Ordering.natural().nullsLast().sortedCopy(numbers); //10, 20,30,60,80,null

Ordering.natural().nullsFirst().sortedCopy(numbers); //null,10,20,30,60,80


public static void testOrdering()
{
List<Person> personList=Lists.newArrayList(
new Person(3, 1, "abc", "46546", 0, 25),
new Person(2, 1, "ab", "46546", 0, 30),
```

```
new Person(5, 1, "ade", "46546",0, 27),
new Person(1, 1, "a", "46546", 1, 20),
new Person(6, 1, "acc", "46546", 1, 29),
new Person(4, 1, "aef", "46546", 1, 50),
new Person(7, 1, "add", "46546",0, 33)
);

Ordering<Person> byAge=new Ordering<Person>() {
@Override
public int compare( Person left, Person right) {
return right.getAge()-left.getAge();
}
};

for(Person p: byAge.immutableSortedCopy(personList))
{
System.out.println(p);
}
}
```

guava在结合部分的API使用记录完毕，希望对广大屌丝有所帮助。

no pays,no gains!

carter.li
关注 - 18
粉丝 - 85
+加关注

8 0

« 上一篇：guava 学习笔记 使用瓜娃（guava）的选择和预判断使代码变得简洁
» 下一篇：Struts2学习之旅一 你妹的，这么多框架，为啥要用struts2?

posted @ 2013-02-05 19:02 carter.li 阅读(87610) 评论(5) 编辑 收藏

评论列表

#1楼 2013-12-25 09:27 conanplus

学习了，谢谢楼主的总结

支持(0) 反对(0)

#2楼 2014-05-03 22:22 java探索者

谢谢楼主总结!

支持(0) 反对(0)

#3楼 2015-05-26 17:20 PearCore

get好东西手下啦

支持(0) 反对(0)

#4楼 2016-01-21 15:58 cmshome1128

我还是看到，他们代码里面有Maps.xxxx我才百度下滴，直接亮瞎我滴钛合金啦。

支持(0) 反对(0)

#5楼 2017-07-05 09:46 rgqancy

原来还有这么好滴的库啊!

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:
· 蔡崇信公益基金会成立：体育应是教育的一部分
· 软银为银行参与移动业务IPO开条件：提供贷款
· 让"子弹"再飞一会儿
· 搜索知名医院却引入民营医院 百度：已下线相关内容
· 被诺贝尔奖忽视的她 刚刚拿下了300万美元大奖
» 更多新闻...

最新知识库文章:
· 如何招到一个靠谱的程序员
· 一个故事看懂"区块链"
· 被踢出去的用户
· 成为一个有目标的学习者
· 历史转折中的"杭派工程师"
» 更多知识库文章...