Google 开源的 Guava 工具库，太强大了 ~

点击关注 Java技术栈 昨天

作者：张丰哲

出处：www.jianshu.com/p/97778b21bd00

目前Google Guava在实际应用中非常广泛，本篇博客将以博主对Guava使用的认识以及在项目中的经验来给大家分享！正如标题所言，学习使用Google Guava可以让你快乐编程，写出优雅的JAVA代码！

以面向对象思想处理字符串:Joiner/Splitter/CharMatcher

> JDK提供的String还不够好么？
>
> 也许还不够友好，至少让我们用起来还不够爽，还得操心！
>
> 举个栗子，比如String提供的split方法，我们得关心空字符串吧，还得考虑返回的结果中存在null元素吧，只提供了前后trim的方法（如果我想对中间元素进行trim呢）。

那么，看下面的代码示例，guava让你不必在操心这些：

```
//连接器
private static final Joiner joiner = Joiner.on(",").skipNulls();

//分割器
private static final Splitter splitter = Splitter.on(",").trimResults().omitEmptyStrings();

public static void main(String[] args) {

    //把集合/数组中的元素join在一起
    String join = joiner.join(Lists.newArrayList("a",null,"b"));
    System.out.println("join=" + join);

    for(String tmp : splitter.split(" a,    ,b,,")){
        System.out.println("|" + tmp + "|");
    }
```

Joiner/Splitter

> Joiner是连接器，Splitter是分割器，通常我们会把它们定义为static final，利用on生成对象后在应用到String进行处理，这是可以复用的。要知道apache commons StringUtils提供的都是static method。
>
> 更加重要的是，guava提供的Joiner/Splitter是经过充分测试，它的稳定性和效率要比apache高出不少，这个你可以自行测试下~
>
> 推荐阅读：试试 StringJoiner，真香!
>
> 发现没有我们想对String做什么操作，就是生成自己定制化的Joiner/Splitter，多么直白，简单，流畅的API!
>
> 对于Joiner，常用的方法是　跳过NULL元素：skipNulls() / 对于NULL元素使用其他替代：useForNull(String)
>
> 对于Splitter，常用的方法是：trimResults()/omitEmptyStrings()。注意拆分的方式，有字符串，还有正则，还有固定长度分割（太贴心了！）

其实除了Joiner/Splitter外，guava还提供了字符串匹配器：CharMatcher

CharMatcher

> CharMatcher，将字符的匹配和处理解耦，并提供丰富的方法供你使用!

对基本类型进行支持

> guava对JDK提供的原生类型操作进行了扩展，使得功能更加强大!

Ints

> guava提供了Bytes/Shorts/Ints/longs/Floats/Doubles/Chars/Booleans这些基本数据类型的扩展支持，只有你想不到的，没有它没有的！

对JDK集合的有效补充

灰色地带:Multiset

> JDK的集合，提供了有序且可以重复的List，无序且不可以重复的Set。那这里其实对于集合涉及到了2个概念，一个order，一个dups。那么List vs Set，and then some ？

Multiset

> Multiset是什么，我想上面的图，你应该了解它的概念了。Multiset就是无序的，但是可以重复的集合，它就是游离在List/Set之间的“灰色地带”！（至于有序的，不允许重复的集合嘛，guava还没有提供，当然在未来应该会提供UniqueList，我猜的，哈哈）

来看一个Multiset的示例：

Multiset Code

> Multiset自带一个有用的功能，就是可以跟踪每个对象的数量。

Immutable vs unmodifiable

来我们先看一个unmodifiable的例子：

unmodifiable

> 你看到JDK提供的unmodifiable的缺陷了吗？Java集合系列面试题我都整理好了，关注公众号Java技术栈回复 "面试" 获取。
>
> 实际上，Collections.unmodifiableXxx所返回的集合和源集合是同一个对象，只不过可以对集合做出改变的API都被override，会抛出UnsupportedOperationException。
>
> 也即是说我们改变源集合，导致不可变视图（unmodifiable View）也会发生变化，oh my god!

当然，在不使用guava的情况下，我们是怎么避免上面的问题的呢？

defensive copies

> 上面揭示了一个概念：Defensive Copies，保护性拷贝。

OK，unmodifiable看上去没有问题呢，但是guava依然觉得可以改进，于是提出了Immutable的概念，来看：

Immutable

> 就一个copyOf，你不会忘记，如此cheap~
>
> 用Google官方的说法是：we're using just one class,just say exactly what we mean，很了不起吗（不仅仅是个概念，Immutable在COPY阶段还考虑了线程的并发性等，很智能的！），O(∩_∩)O哈哈~
>
> guava 提供了很多Immutable集合，比如ImmutableList/ImmutableSet/ImmutableSortedSet/ImmutableMap/.....

看一个ImmutableMap的例子：

ImmutableMap

可不可以一对多：Multimap

> JDK提供给我们的Map是一个键，一个值，一对一的，那么在实际开发中，显然存在一个KEY多个VALUE的情况（比如一个分类下的书本），我们往往这样表达：`Map<k,List<v>>`，好像有点臃肿！臃肿也就算了，更加不爽的事，我们还得判断KEY是否存在来决定是否new 一个LIST出来，有点麻烦！更加麻烦的事情还在后头，比如遍历，比如删除，so hard.....

来看guava如何替你解决这个大麻烦的：

Multimap

> 友情提示下，guava所有的集合都有create方法，这样的好处在于简单，而且我们不必在重复泛型信息了。
>
> get()/keys()/keySet()/values()/entries()/asMap() 都是非常有用的返回view collection的方法。
>
> Multimap的实现类有：ArrayListMultimap/HashMultimap/LinkedHashMultimap/TreeMultimap/ImmutableMultimap/……

可不可以双向：BiMap

> JDK提供的MAP让我们可以find value by key，那么能不能通过find key by value呢，能不能KEY和VALUE都是唯一的呢。这是一个双向的概念，即forward+backward。
>
> 在实际场景中有这样的需求吗？比如通过用户ID找到mail，也需要通过mail找回用户名。没有guava的时候，我们需要create forward map AND create backward map，and now just let guava do that for you.

BiMap

> biMap / biMap.inverse() / biMap.inverse().inverse() 它们是什么关系呢?
>
> 你可以稍微看一下BiMap的源码实现，实际上，当你创建BiMap的时候，在内部维护了2个map，一个forward map，一个backward map，并且设置了它们之间的关系。
>
> 因此，biMap.inverse() != biMap ； biMap.inverse().inverse() == biMap

可不可以多个KEY：Table

> 我们知道数据库除了主键外，还提供了复合索引，而且实际中这样的多级关系查找也是比较多的，当然我们可以利用嵌套的Map来实现：`Map<k1,Map<k2,v2>>`。为了让我们的代码看起来不那么丑陋，guava为我们提供了Table。

Table

> Table涉及到3个概念：rowKey,columnKey,value，并提供了多种视图以及操作方法让你更加轻松的处理多个KEY的场景。

函数式编程：Functions

Functions

> 上面的代码是为了完成将List集合中的元素，先截取5个长度，然后转成大写。
>
> 函数式编程的好处在于在集合遍历操作中提供自定义Function的操作，比如transform转换。我们再也不需要一遍遍的遍历集合，显著的简化了代码！

对集合的transform操作可以通过Function完成

断言：Predicate

> Predicate最常用的功能就是运用在集合的过滤当中！

filter

> 需要注意的是Lists并没有提供filter方法，不过你可以使用Collections2.filter完成！推荐阅读：Java 中初始化 List 集合的 6 种方式。

check null and other：Optional、Preconditions

在guava中，对于null的处理手段是快速失败，你可以看看guava的源码，很多方法的第一行就是：Preconditions.checkNotNull(elements);

要知道null是模糊的概念，是成功呢，还是失败呢，还是别的什么含义呢?

Preconditions/Optional

Cache is king

> 对于大多数互联网项目而言，缓存的重要性，不言而喻!
>
> 如果我们的应用系统，并不想使用一些第三方缓存组件（如redis），我们仅仅想在本地有一个功能足够强大的缓存，很可惜JDK提供的那些SET/MAP还不行!

CacheLoader

> 首先，这是一个本地缓存，guava提供的cache是一个简洁、高效，易于维护的。为什么这么说呢？因为并没有一个单独的线程用于刷新 OR 清理cache，对于cache的操作，都是通过访问/读写带来的，也就是说在读写中完成缓存的刷新操作！
>
> 其次，我们看到了，我们非常通俗的告诉cache，我们的缓存策略是什么，SO EASY！在如此简单的背后，是guava帮助我们做了很多事情，比如线程安全。

让异步回调更加简单

> JDK中提供了Future/FutureTask/Callable来对异步回调进行支持，但是还是看上去挺复杂的，能不能更加简单呢？比如注册一个监听回调。

异步回调

> 我们可以通过guava对JDK提供的线程池进行装饰，让其具有异步回调监听功能，然后在设置监听器即可！

Summary

到这里，这篇文章也只介绍了guava的冰山一角，其实还有很多内容：

guava package

> 比如反射、注解、网络、并发、IO等等

好了，希望这篇文章让你快速进阶，快乐编程！

关注Java技术栈看更多干货

戳原文，获取精选面试题！

阅读原文

喜欢此内容的人还喜欢

刚来的大兄弟在这个小问题上翻车了，你确定不看一下？

故里学Java

涨姿势了！delete后加 limit是个好习惯么？

Java大后端

Redis：从应用到底层，一文帮你搞定

sowhat1412