Guava库介绍之实用工具类

作者:Jack47

欢迎关注我的微信公众账号程序员杰克，两边的文章会同步，也可以添加我的RSS订阅源。

本文是我写的Google开源的Java编程库Guava系列之一，主要介绍Guava中提供的很多小工具类，这些类让Java语言用起来更舒畅。

使用或者避免null值

`null` 引用的发明者Sir C.A.R.Hoare(也是快排算法的发明者)把 `null` 称之为十亿美元错误。 `Guava` 的开发者们通过研究Google的代码发现95%的集合中都不需要支持为 `null` 的值，所以对于开发者而言，遇到 `null` 时需要快速失败而不是默默地接受 `null` 。 `null` 的含义在大部分场景下都不够清晰，例如 `Map.get(key)` 返回 `null` 时，可能是因为map中的值就是 `null` ，或者map中没有这个key。但 `null` 在有些情况下也很有用，从内存和性能方面来看， `null` 很廉价，在使用对象数组，是不可避免的要用到 `null` 的。综合考虑之后，Guava库中绝大部分工具都被设计成遇到 `null` 时快速失败。

Optional

如果程序员需要使用 `null` 来表示不存在的情况，那么 `Optional<T>` 就能派上用场了。 `Optional<T>` 是用非 `null` 的值来代替一个可能为 `null` 的值。举个例子：

```
Optional<Integer> possible = Optional.of(null);
boolean isPresent = possible.isPresent(); // returns false
```

再看这个例子:

```
Optional<Integer> possible = Optional.of(5);
boolean isPresent = possible.isPresent(); // returns true
possible.get(); // returns 5
```

聪明的读者已经发现了， `Optional` 是一个容器对象，它可能容纳一个非 `null` 的值，也可能没有值。如果这个值存在， `isPresent()` 函数会返回 `true` 而且 `get()` 函数会返回这个值。

引入 `Optional` 类除了因为给 `null` 一个有意义的名字而增加了可读性外，最大的好处是 `Optional` 能够强制你主动思考程序中值不存在的情况，而 `null` 是很容易被忽略的。如果值不存在，想使用默认值，可以使

导航

首页
新随笔
联系
订阅XML
管理

< 2018年5月 >

日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

用 `Optional` 中提供的 `T or(T)` 函数，例如：

```
Optional<Integer> default = Optional.of(0);
Optional possible = Optional.of(field);
return possible.or(default)
```

如果field为 `null` ,那么返回值就是 `default` ，调用者需要先判断值是否存在 `isPresent()` ，如果存在，再拿到值 `get()` 。

`Strings` 类提供了值可能为 `null` 的一些函数，例如：

```
boolean invalid = Strings.isNullOrEmpty(res);
String name = Strings.emptyToNull(passedName);
```

Preconditions(先决条件)

`Preconditions` 类中提供了几个非常实用的静态函数来帮助检查调用函数或构造函数时的先决条件是否满足。这些函数都接受一个 `boolean` 类型的表达式，如果表达式为 `false` ,会抛出一个非受检异常，来通知调用方发生了调用错误。

`Preconditions` 类的目的是提高代码的可读性，例如 `checkArgument(i >=0 , "Argument was %s but expected nonnegative", i);` ，一看就知道是在检查传入的参数； `this.field = checkNotNull(field)` ，是检查field字段不为空。这里需要注意提供的错误信息需要清晰有效。

对象的通用函数

使用这些函数能够简化实现对象函数的过程，例如 `hashCode()` 和 `toString()` 函数

equal函数

如果对象内部变量可以为 `null` ,实现 `equal` 函数有些费劲，因为需要单独检查 `null` 。
`Objects.equal` 函数已经实现了对 `null` 敏感的检查，不会出现 `NullPointerException` 的异常。

```
Objects.equal("a", "a"); //returns true
Objects.equal(null, "a"); // returns false
Objects.equal(null, null); // returns true
```

注：最新的JDK 7里引入了 `Ojbects` 类，提供了同样的函数。

hashCode函数

平常实现hashCode函数是不是很痛苦？如果类内部的成员变量较多，代码就会比较冗长。Guava提供了 `Objects.hashCode(field1, field2,...,fieldn)` 的函数，它能够对指定顺序的多个字段进行哈希，这样就不用自己手工实现一遍对各个字段进行哈希的过程了。举个栗子：

```
return Objects.hashCode(name, address, url);
```

toString函数

`toString` 在日志和调试中发挥巨大威力，但是实现起来很麻烦，需要注意各个有用字段输出时的组织格式。来看看 `MoreObjects.toStringHelper` 如何让整个过程变的简单：

```
// return "MyObject{x=1,y=2}
Objects.toStringHelper(this)
        .add("x", 1)
        .add("y", 2)
        .toString();
```

compare/compareTo函数

实现比较器(Comparator)，或者实现 `Comparable` 接口时，通常需要对类内部的所有成员变量进行比较，实现起来很麻烦。Guava提供了 `ComparisonChain` 类，它能够进行"懒"比较：只有当发现为0的结果，才会继续后面的结果，不然就忽略后续的比较。举个例子：

```
public int compareTo(Foo other) {
    return ComparisonChain.start()
    .compare(this.x, other.x)
    .compare(this.y, other.y)
    .result();
}
```

Ordering

Guava提供了一个流畅型(fluent)的比较器(Comparator)类：`Ordering` ，它提供了链条函数来微调或者增强已有的比较器，或者构造复杂的比较器，应用到对象的集合上。

`Ordering` 的核心是一个 `Comparator` 实例。使用已有的比较器来构造一个 `Ordering` 实例：

```
Ordering<Item> ordering = Ordering.from(Comparator<Item> comparator);
```

也可以使用自然顺序： `Ordering<T>.natural()`

或者自己实现一个 `Ordering` 类，只需要继承并实现 `compare()` 函数就可以。

对 `Ordering` 进行微调：

```
reverse()
compound(Comparator)
onResultOf(Function)
nullsFirst()
```

由于 `Ordering` 类继承自 `Comparator` ,所以在任何需要 `Comparator` 的地方，都可以使用 `Ordering` 代替，同时 `Ordering` 提供了一些操作：

```
immutableSortedCopy()
isOrdered()/ isStrictlyOrdered()
min()
```

字符串相关的函数

技术博客链接

酷壳 – 享受编程和技术所带来的快乐
刘未鹏 | Mind Hacks 思维改变生活

最新评论

1. Re:从现在起入门机器学习
支持支持
--牛腩

2. Re:Storm介绍(一)
@luchi123贴一下github上的源码在哪里？...
--Jack47

3. Re:Storm介绍(一)
@Jack47找不到这个方法 也不清楚这个方法代表的含义 这些理论化的东西我在网上翻烂了 Utils类的waitForMills（1）方法 这个方法 有问题 我讲的.....
--luchi123

4. Re:基于Kubernetes在AWS上部署Kafka时遇到的一些问题
@yew1eb文中提到了，就是那个配置项：...
--Jack47

5. Re:Storm介绍(一)
@luchi123我怎么回答呢？具体什么错误？上google搜一下这个错误？...
--Jack47

6. Re:Storm介绍(一)
楼主 我想问一下storm的简单例子中有一个Utils类的waitForMills（1）方法 但是自己单机敲的时候 无法点出这个方法 ide报错
--luchi123

合并(Joiner)

流畅风格的 `Joiner` 提供了使用分隔符把一些字符串序列合并到一起的功能。例如：

```
Joiner joiner = Joiner.on("; ").skipNulls();
// returns "Harry; Ron; Hermione
return joiner.join("Harry", null, "Ron", "Hermione");
```

如果不想跳过值为 `null` 的字符串，想用某个特定字符串代替 `null` ,可以使用函数 `useForNull(String)` 。
`Joiner` 类也可以用在其他对象类型上，此时会使用对象的 `toString()` 函数得到字符串，然后进行合并。

切分(Splitter)

Java的字符串分割函数有一些诡异的行为，例如 `String.split()` 函数会默默地把尾部的分隔符丢弃掉。而使用 `Splitter` 的好处在于可以完整地显示地控制这些行为。 `Splitter` 类可以用来在任意的 `Pattern` , `char` , `String` 或者 `CharMatcher` 上进行分割。举个例子：

```
// returns ["foo", " bar", "", "   qux", ""]
Splitter.on(',').split("foo, bar,,   qux,");
```

`Splitter` 还提供了其他的配置函数来对合并过程进行定制： `omitEmptyStrings()` , `trimResults()` , `limit()` 等。例如对于上面的例子，如果想忽略空字符串，让结果中去掉开头和结尾的空格，可以这样做：

```
// returns ["foo","bar","qux"]
Splitter.on(',')
                .trimReults()
                .omitEmptyStrings()
                .split("foo, bar,,   qux,");
```

注意： `Splitter` 和 `Joiner` 实例都是不可变的(immutable)，所以 `Splitter` 和 `Joiner` 都是线程安全的，可以声明为 `static final` 的常量来使用。他们的配置函数都是返回一个新的 `Splitter` 实例，此时需要使用返回的新的 `Splitter` 的实例。

字符匹配(Character Matchers)

`CharMatcher` 类的设计思想很巧妙，定义两个基本属性，然后任意组合他们。这样API的复杂度是线性增加的，但是灵活性和功能是平方式增强的。

`CharMatcher` 定义的两个属性：

1. 需要匹配什么样的字符串？
2. 在这些匹配的字符串上执行什么样的操作？例如 `trimming` , `collapsing` , `removing` 等。

一些例子：

```
    // remove control characters
    String noControl = CharMatcher.JAVA_ISO_CONTROL.removeFrom(inputString);
    // only the digits
```


7. Re:基于Kubernetes在AWS上部署Kafka时遇到的一些问题

请问怎么配置每个broker对外暴露的IP呢？

--yew1eb

8. Re:Storm介绍(一)

@Ryan.Miao我只看到了设置-》博客皮肤里是"kubrick" 这个，不知道是不是你想要的？文章的url可以自己设置的，发博客时，“高级选项”-》EntryName里填上文字就可以了。……

--Jack47

9. Re:Storm介绍(一)

楼主的主题是哪个啊，简介大方。而且，您的文章路由是怎么修改的

--Ryan.Miao

10. Re:Kubernetes环境下的各种调试方法

Useful commands for docker beginners

--黄百鸣灵

阅读排行榜

1. Google软件构建工具Bazel原理及使用方法介绍(21075)
2. Linux下服务器端开发流程及相关工具介绍(C++)(15926)
3. Storm介绍(一)(12436)
4. Linux环境下shell和vim中乱码原因及消除办法(11981)
5. Redhat环境下编译安装Google Bazel(11472)
6. Google分布式构建软件之一：获取源代码(7660)
7. [翻译]禅与文件和文件夹组织的艺术 上(6922)
8. Google软件构建工具Bazel FAQ(5517)
9. Google分布式构建软件之二：构建系统如何工作(5156)
10. 为什么google bazel构建工具流行不起来(5037)

评论排行榜

1. Linux下服务器端开发流程及相关工具介绍(C++)(23)
2. [翻译]禅与文件和文件夹组织的艺术 上(8)
3. C语言中TMin的写法(6)

```
    String theDigits = CharMatcher.DIGIT.retainFrom(inputString);
    // eliminate all characters that aren't digits or lowercase
    String lowerAndDigit =
CharMatcher.JAVA_DIGIT.or(CharMatcher.JAVA_LOWER_CASE).retainFrom(inputString);
    // trim whitespace and replace/collapse whitespace into single spaces
    String spaced = CharMatcher.WHITESPACE.trimAndCollapseFrom(inputString, ' ');
```

认真的读者通过看上面的例子会发现 `CharMatcher` 已经提供了很多现成的匹配特定字符串的常量，例如 `WHITESPACE` ，`JAVA_DIGIT` 等。也可以通过其他几个函数来构造匹配特定字符串的 `CharMatcher` ：

```
is(char);
isNot(char);
negate()
inRange(char, char)
or(CharMatcher);
and(CharMatcher);
```

可以在CharMatcher上执行的操作

```
boolean matchesAllOf(CharSequence)
boolean matchesAnyOf(CharSequence)
boolean matchesNoneOf(CharSequence)
int indexIn(CharSequence, int)
int countIn(CharSequence)
String removeFrom(CharSequence)
String retainFrom(CharSequence)
String trimFrom(CharSequence)
String replaceFrom(CharSequence, char)
```

Charsets

`Charsets` 类提供了Java平台的所有实现中都支持的六个标准的字符集的常量引用。所以不要这样做：

```
try {
    bytes = content.getBytes("UTF-8");
} catch (UnsupportedEncodingException e) {
    throw new AssertionError(e);
}
```

而是用下面的代码替代：

```
bytes = content.getBytes(Charsets.UTF_8);
```

注：JDK7中 `StandardCharsets` 类已经实现了同样功能

基本类型相关的函数

Java中 `Arrays` 类提供了众多对数组进行操作的函数，基础类型对应的包装类例如 `Integer` ，也提供了很多使用函数。而 `Guava` 为Java中的8个基本类型提供了其他一些非常实用的函数，例如数组和集合相关的API，从类型转换到 `byte` 数组的表示方式等。

例如：

4. C语言中为什么不能用char类型来存储getchar()的返回值(6)
5. Vim新手入门资料和一些Vim实用小技巧(6)

```
int[] content = {1,3,4};
Ints.indexOf(content, 3); // 1
Ints.concat(new int[] {1,2}, new int[]{3, 4})  // 1,2,3,4
Ints.contains(new int[]{10,20,30,40}, 20) // true
Ints.min(10,20,30,40) // 10
byte[] bytes = Ints.toByteArray(integer);
```

回到本系列目录：Google Java编程库Guava介绍

如果您看了本篇博客,觉得对您有所收获，请点击右下角的“推荐”，让更多人看到！

资助Jack47写作，打赏一个鸡蛋灌饼钱吧

¥4.99

微信打赏

支付宝打赏

标签: Java, API, Guava

Jack47
关注 - 3
粉丝 - 63
+加关注

1 0

« 上一篇：Google Java编程库Guava介绍
» 下一篇：Storm介绍(二)

posted on 2016-02-29 06:34 Jack47 阅读(2110) 评论(0) 编辑 收藏

站长统计