


导航

博客园
首 页
新随笔
联 系
订 阅 
管 理

< 2018年9月 >						
日	一	二	三	四	五	六
26	27	28	29	30	31	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	1	2	3	4	5	6

公告

昵称: 乌云上
园龄: 1年9个月
粉丝: 1
关注: 6
[+加关注](#)

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论

guava快速入门（一）

Guava工程包含了若干被Google的 Java项目广泛依赖 的核心库，例如：集合 [collections]、缓存 [caching]、原生类型支持 [primitives support]、并发库 [concurrency libraries]、通用注解 [common annotations]、字符串处理 [string processing]、I/O 等等。

guava类似Apache Commons工具集

基本工具包Base

Optional

guava的Optional类似于Java 8新增的Optional类，都是用来处理null的，不过guava的是抽象类，其实现类为Absent和Present，而java.util的是final类。其中一部分方法名是相同的。

Guava用Optional表示可能为null的T类型引用。一个Optional实例可能包含非null的引用（我们称之为引用存在），也可能什么也不包括（称之为引用缺失）。它从不说包含的是null值，而是用存在或缺失来表示。但Optional从不会包含null值引用。

```
1  import com.google.common.base.Optional;
2
3  /**
4   * Guava用Optional表示可能为null的T类型引用。
5   *
6   * @author LENOVO
7   *
8   */
9  public class OptionalDemo {
10
11      public static void main(String[] args) {
12          Integer value1 = null;
13          Integer value2 = 10;
14          /*
15           * 创建指定引用的Optional实例，若引用为null则快速失败返回absent() absent()创建引用缺失的Optional实例
```

[我的标签](#)

我的标签

[日常积累\(110\)](#)[Java\(26\)](#)[数据库\(24\)](#)[SqlServer\(22\)](#)[环境搭建\(21\)](#)[SQL\(19\)](#)[jQuery\(18\)](#)[php\(16\)](#)[Node.js学习\(15\)](#)[JavaScript\(15\)](#)[更多](#)

随笔分类

[.NET\(6\)](#)[Ajax\(1\)](#)[BAT\(1\)](#)[CASCO卡斯柯](#)[Dos命令\(1\)](#)[Excel\(2\)](#)[Game\(1\)](#)[Java\(27\)](#)[jQuery\(17\)](#)[jQuery easyUI\(1\)](#)[Node.js\(16\)](#)[PHP\(14\)](#)[Python\(15\)](#)[SQL\(26\)](#)[WeUI\(10\)](#)[总结\(4\)](#)

随笔档案

[2018年9月 \(19\)](#)[2018年8月 \(54\)](#)[2018年7月 \(6\)](#)[2018年6月 \(43\)](#)[2018年5月 \(13\)](#)[2018年4月 \(6\)](#)[2018年3月 \(1\)](#)[2018年1月 \(1\)](#)

积分与排名

```

16     */
17     Optional<Integer> a = Optional.fromNullable(value1);
18     Optional<Integer> b = Optional.of(value2); // 返回包含给定的非空引用Optional实例
19     System.out.println(sum(a, b));
20 }
21
22 private static Integer sum(Optional<Integer> a, Optional<Integer> b) {
23     // isPresent():如果Optional包含非null的引用（引用存在），返回true
24     System.out.println("First param is present: " + a.isPresent());
25     System.out.println("Second param is present: " + b.isPresent());
26     Integer value1 = a.or(0); // 返回Optional所包含的引用,若引用缺失,返回指定的值
27     Integer value2 = b.get(); // 返回所包含的实例,它必须存在,通常在调用该方法时会调用isPresent()判断是否为null
28     return value1 + value2;
29 }
30 }

```

运行返回:

```

1 First param is present: false
2 Second param is present: true
3 10

```

Preconditions

前置条件Preconditions提供静态方法来检查方法或构造函数，被调用是否给定适当的参数。它检查的先决条件。其方法失败抛出IllegalArgumentExpection。

```

1 import com.google.common.base.Preconditions;
2
3 /**前置条件Preconditions提供静态方法来检查方法或构造函数
4  * @author LENOVO
5  *
6  */
7 public class PreconditionDemo {
8
9     public static void main(String[] args) {
10         try {

```

积分 - 5839

排名 - 58324

最新评论

1. Re:如何把ASP.NET MVC
项目部署到本地IIS上

@~雨落忧伤~引用别人能访问吗? 可以的, 前提是, 访问者和你的电脑在同一个局域网, 并且知道你的电脑ip地址, 比如, 你用手机连上你电脑开出来的wifi, 然后再手机的浏览器中输入ip地址也是可以访问的。.....

--乌云上

2. Re:如何把ASP.NET MVC
项目部署到本地IIS上

别人能访问吗?

--~雨落忧伤~

阅读排行榜

1. 常见浏览器User-Agent
大全(773)

2. Python——第一个
python程序
helloworld(654)

3. jquery里判断数组内是否
包含了指定的值或元素的方法(368)

4. 如何把ASP.NET MVC项
目部署到本地IIS上(329)

5. jquery操作radio单选按钮, 实现取值, 动态选中, 动态删除的各种方法(328)

评论排行榜

1. 如何把ASP.NET MVC项
目部署到本地IIS上(2)

```

11         getValue(5);
12     } catch (IndexOutOfBoundsException e){
13         System.out.println(e.getMessage());
14     }
15
16     try {
17         sum(4,null);
18     } catch (NullPointerException e){
19         System.out.println(e.getMessage());
20     }
21
22     try {
23         sqrt(-1);
24     } catch (IllegalArgumentException e){
25         System.out.println(e.getMessage());
26     }
27
28 }
29
30 private static double sqrt(double input){
31     Preconditions.checkArgument(input>0.0,
32         "Illegal Argument passed: Negative value %s.",input);
33     return Math.sqrt(input);
34 }
35
36 private static int sum(Integer a,Integer b){
37     a=Preconditions.checkNotNull(a,
38         "Illegal Argument passed: First parameter is Null.");
39     b=Preconditions.checkNotNull(b,
40         "Illegal Argument passed: Second parameter is Null.");
41     return a+b;
42 }
43
44 private static int getValue(int input){
45     int[] data={1,2,3,4,5};
46     int index=Preconditions.checkNotNull(input,data.length,
47         "Illegal Argument passed: Invalid index.");
48     return data[index];

```

```
49     }  
50  
51 }
```

返回:

```
1 Illegal Argument passed: Invalid index. (5) must be less than size (5)  
2 Illegal Argument passed: Second parameter is Null.  
3 Illegal Argument passed: Negative value -1.0.
```

Joiner

Joiner 提供了各种方法来处理字符串加入操作，对象等。

Joiner的实例不可变的，因此是线程安全的。

Warning: joiner instances are always immutable; a configuration method such as { useForNull} has no effect on the instance it is invoked on! You must store and use the new joiner instance returned by the method. This makes joiners thread-safe, and safe to store as {@code static final} constants.

```
{@code  
  
    // Bad! Do not do this!  
  
    Joiner joiner = Joiner.on(',');  
  
    joiner.skipNulls(); // does nothing!分开写跳过null就不起作用了，因为实例不可改变  
  
    return joiner.join("wrong", null, "wrong");}
```

```
1 import java.util.Arrays;  
2 import java.util.HashMap;  
3 import java.util.Map;  
4
```

```
5 import com.google.common.base.Joiner;
6
7 /**Joiner 提供了各种方法来处理字符串加入操作，对象等
8  * @author LENOVO
9  *
10  */
11 public class JoinerDemo {
12
13     public static void main(String[] args) {
14         /*
15          * on:制定拼接符号，如：test1-test2-test3 中的“-“ 符号
16          * skipNulls(): 忽略NULL,返回一个新的Joiner实例
17          * useForNull(“Hello”): NULL的地方都用字符串“Hello”来代替
18          */
19         StringBuilder sb=new StringBuilder();
20         Joiner.on(",").skipNulls().appendTo(sb,"Hello","guava");
21         System.out.println(sb);
22         System.out.println(Joiner.on(",").useForNull("none").join(1,null,3));
23         System.out.println(Joiner.on(",").skipNulls().join(Arrays.asList(1,2,3,4,null,6)));
24         Map<String,String>map=new HashMap<>();
25         map.put("key1","value1");
26         map.put("key2","value2");
27         map.put("key3","value3");
28         System.out.println(Joiner.on(",").withKeyValueSeparator("=").join(map));
29     }
30 }
```

返回：

```
1 Hello,guava
2 1,none,3
3 1,2,3,4,6
4 key1=value1,key2=value2,key3=value3
```

Splitter

Splitter 能够将一个字符串按照指定的分隔符拆分成可迭代遍历的字符串集合，Iterable

```
1 import com.google.common.base.Splitter;
2
3 /**Splitter 能够将一个字符串按照指定的分隔符拆分成可迭代遍历的字符串集合
4  * @author LENOVO
5  *
6  */
7 public class SplitterDemo {
8
9     public static void main(String[] args) {
10
11         /*
12             on():指定分隔符来分割字符串
13             limit():当分割的子字符串达到了limit个时则停止分割
14             fixedLength():根据长度来拆分字符串
15             trimResults():去掉子串中的空格
16             omitEmptyStrings():去掉空的子串
17             withKeyValueSeparator():要分割的字符串中key和value间的分隔符,分割后的子串中key和value间的分隔符默认是=
18         */
19         System.out.println(Splitter.on(",").limit(3).trimResults().split(" a, b, c, d")); //[ a, b, c,d]
20         System.out.println(Splitter.fixedLength(3).split("1 2 3")); //[1 2, 3]
21         System.out.println(Splitter.on(" ").omitEmptyStrings().splitToList("1 2 3"));
22         System.out.println(Splitter.on(",").omitEmptyStrings().split("1,,,2,,,3")); //[1, 2, 3]
23         System.out.println(Splitter.on(" ").trimResults().split("1 2 3")); //[1, 2, 3],默认的连接符是,
24         System.out.println(Splitter.on(";").withKeyValueSeparator(":").split("a:1;b:2;c:3")); //{a=1, b=2, c=3}
25     }
26 }
```

返回:

```
1 [a, b, c, d]
2 [1 2, 3]
3 [1, 2, 3]
4 [1, 2, 3]
5 [1, 2, 3]
6 {a=1, b=2, c=3}
```

Objects

java7及以后的版本建议使用jdk中的Objects类

EventBus

Guava为我们提供了事件总线EventBus库，它是事件发布-订阅模式的实现，让我们能在领域驱动设计(DDD)中以事件的弱引用本质对我们的模块和领域边界很好的解耦设计。

Guava为我们提供了同步事件EventBus和异步实现AsyncEventBus两个事件总线，他们都不是单例的。

Guava发布的事件默认不会处理线程安全的，但我们可以标注@AllowConcurrentEvents来保证其线程安全

如果Listener A监听Event A, 而Event A有一个子类Event B, 此时Listener A将同时接收Event A和B消息

事件

```
1  /**Guava 发布-订阅模式中传递的事件,是一个普通的POJO类
2   * @author LENOVO
3   *
4   */
5  public class OrderEvent { // 事件
6      private String message;
7
8      public OrderEvent(String message) {
9          this.message = message;
10     }
11
12     public String getMessage() {
13         return message;
14     }
15 }
```

订阅

```
1  import com.google.common.eventbus.Subscribe;
2
3  public class EventListener { // 订阅者
4
5      // @Subscribe保证有且只有一个输入参数,如果你需要订阅某种类型的消息,只需要在指定的方法上加上@Subscribe注解即可
6      @Subscribe
7      public void listen(OrderEvent event) {
```

```
8         System.out.println("receive message: " + event.getMessage());
9     }
10
11     /*
12      * 一个subscriber也可以同时订阅多个事件 Guava会通过事件类型来和订阅方法的形参来决定到底调用subscriber的哪个订阅方法
13      */
14     @Subscribe
15     public void listen(String message) {
16         System.out.println("receive message: " + message);
17     }
18 }
```

多个订阅者

```
1 import com.google.common.eventbus.Subscribe;
2
3 public class MultiEventListener {
4
5     @Subscribe
6     public void listen(OrderEvent event) {
7         System.out.println("receive msg: " + event.getMessage());
8     }
9
10    @Subscribe
11    public void listen(String message) {
12        System.out.println("receive msg: " + message);
13    }
14 }
```

Demo:

```
1 import com.google.common.eventbus.EventBus;
2
3 public class EventBusDemo {
4
5     public static void main(String[] args) {
6         EventBus eventBus = new EventBus("jack");
7         /*
8          * 如果多个subscriber订阅了同一个事件,那么每个subscriber都将收到事件通知
9          */
10    }
```



```
9      * 并且收到事件通知的顺序跟注册的顺序保持一致
10     */
11     EventBus.register(new EventListener()); // 注册订阅者
12     EventBus.register(new MultiEventListener());
13     EventBus.post(new OrderEvent("hello")); // 发布事件
14     EventBus.post(new OrderEvent("world"));
15     EventBus.post("!"); // 只有他才会执行所有的事件
16 }
17
18 }
```

返回:

```
1 receive message: hello
2 receive msg: hello
3 receive message: world
4 receive msg: world
5 receive message: !
6 receive msg: !
```

DeadEvent

如果EventBus发送的消息都不是订阅者关心的称之为Dead Event。

```
1 import com.google.common.eventbus.DeadEvent;
2 import com.google.common.eventbus.Subscribe;
3
4 public class DeadEventListener {
5     boolean isDelivered = true;
6
7     @Subscribe
8     public void listen(DeadEvent event) {
9         isDelivered = false;
10        System.out.println(event.getSource().getClass() + " " + event.getEvent()); // source通常是EventBus
11    }
12
13    public boolean isDelivered() {
```

```
14         return isDelivered;
15     }
16 }
```

参考: <http://www.importnew.com/tag/guava>

分类: [Java](#)

标签: [日常积累](#), [Java](#), [guava](#), [guava基础](#)

好文要顶

关注我

收藏该文



[乌云上](#)

关注 - 6

粉丝 - 1

[+加关注](#)

« 上一篇: [guava文档API制作成chm文件](#)

» 下一篇: [13个开发者技能必知必会!](#)

posted on 2018-06-20 15:16 乌云上 阅读(42) 评论(0) 编辑 收藏

[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:**

- 约谈整改28天后探访：北京租房仍存虚假房源 中介自爆商业内幕
 - 华为手机芯片麒麟980发布：投入3亿美元 用7纳米工艺
 - 太空殖民拯救人类？灭绝可能性或上升 还需三思而行
 - 二手车交易平台乱象丛生：隐形高收费 刷单造繁荣
 - 谷歌拒派最高领导参加听证会 美参议院：深感失望
- » 更多新闻...



华为全联接大会 | 上海 | 2018.10.10-12

[大会门票+云服务器]专属套餐0.35折起

**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...

Powered by:

博客园

Copyright © 乌云上