Guava库学习:函数编程(一)使用Function和Functions进行对象转换

链接地址：http://www.xx566.com/detail/130.html

在了解和学习Guava中Function和Functions之前，我们需要先来简单的了解一下函数式编程，函数编程是一种编程范型，强调使用函数来实 现对象不断变化的状态，其本质是一种编程方法或编程思想，著名的函数式编程语言有：Lisp、Erlang、Clojure、Scala等，在JDK8之 前，java只能通过匿名类来达到近似函数式编程的效果，不过JDK8已经增加了函数式编程的支持，而使用Guava，我们可以在JDK5以上，就使用 Java进行函数式编程。

在Guava中，提供了多种接口或类，来进行函数式编程，我们可以使用Function和Functions进行对象转换，使用Predicate和 Predicates进行对象过滤，使用Supplier和Suppliers进行对象的包装构建等，我们首先学习Function和Functions 进行对象转换。

在Java开发中，我们经常需要对一些对象做一些处理，然后返回我们需要看的结果，比如说：对日期进行格式化，获取字符串表示等，当然，我们可以使用String的Formatter来处理（详见：JDK1.5中java.util.Formatter类的学习探究和使用），不过在Guava中我们可以使用Function接口来实现类似的需求，如下：

```
@Test
public void testFunction() {
    Function<Date, String> function = new Function<Date, String>() {
        @Override
        public String apply(Date input) {
```

```
            return new SimpleDateFormat("yyyy-MM-dd").format(input);
        }
    };
    System.out.println(function.apply(new Date()));//2014-08-21
}
```

我们看到，Function接口中的apply方法，接收input参数，返回处理后的结果，我们可以通过调用apply方法，获取想要的结果，其实做过 Android的，会感觉上面的很熟悉，我们在Android开发中，一些事件的处理，编码都类似于上面的风格，这就是所谓的函数式编程。

翻开Function接口的源码，里面只有简单的apply和equals两个方法，我们重点来看apply方法，apply方法接收一个输入对象并返回 一个输出 对象。一个好的Function实现应该没有副作用，也就是说对象作为参数传递方法调用apply方法后应保持不变。

多数情况下，我们都需要自己书写Function接口的实现，以适应工作的需求，不过Guava提供了Functions工具类，其包含了一些常用的Fu nction实现，翻开Functions源码，我们看到有以下几个方法，值得注意的是，其中大量使用了枚举的单例实现【更多：三种方式实现类的Singleton(单例)】，这里不再赘述，其中公有的方法如下：

toStringFunction()：返回ToStringFunction的唯一实例，ToStringFunction主要是对返回传入对象调用toString方法后的表示。

identity()：返回IdentityFunction的唯一实例。

forMap(Map<K, V> map)，forMap(Map<K, ? extends V> map, @Nullable V defaultValue)：接收一个Map集合作为参数，返回一个Function，用于执行Map集合的查找。

compose(Function<B, C> g, Function<A, ? extends B> f)：接收两个Function作为参数，返回两个Function的组合。

forPredicate(Predicate<T> predicate)：接收Predicate，转变为Function，更多Predicate，参见【Guava库学习：函数编程（二）使用Predicate和Predicates进行对象过滤】

constant(@Nullable E value)：为任何输入的值构造一个Function

forSupplier(Supplier<T> supplier)：接收Supplier，转变为Function，更多Supplier，参见【Guava库学习：函数编程（三）使用Supplier和Suppliers进行对象的包装构建】

接下来，我们通过实例代码，学习其中的两个重要的方法，forMap和compose方法，如下：

```
    @Test
    public void testFunctions() {
        Map<String, Integer> map = new HashMap<String, Integer>() {
            //构造一个测试用Map集合
            {
                put("love", 1);
                put("miss", 2);
            }
        };
        /**
         * forMap
         */
        Function<String, Integer> function = Functions.forMap(map);
        //调用apply方法，可以通过key获取相应的value
        System.out.println(function.apply("love"));//i love u
        //当apply的key值在Map里不存在时，会抛出异常
        //java.lang.IllegalArgumentException: Key 'like' not present in map
//        System.out.println(function.apply("like"));
        //我们可以通过forMap重载的另一个方法避免异常，当Key不存在时，设置一个默认值
        function = Functions.forMap(map, 0);
        System.out.println(function.apply("like"));//can't find this key
        /**
         * 有时候，我们需要多个Function进行组合，
         * 这时就需要用到compose，如下：
         */
        //我们有一个Function用于将输入的数字进行平方运算
        Function<Integer, Integer> function1 = new Function<Integer, Integer>() {
            @Override
            public Integer apply(Integer input) {
                return input * input;
            }
        };
        //我们将上面Map中的value值全部进行平方运算
```

```
        /**
         * Warning：这里compose方法的参数位置不能颠倒，
         * Function<A, C> compose(Function<B, C> g, Function<A, ? extends B> f)
         * 传入Function<B,C>、Function<A, ? extends B>组合成Function<A, C>
         */
        Function<String, Integer> result = Functions.compose(function1, function);
        System.out.println(result.apply("love"));//I LOVE U
        //当Key值不存在时，结果也是大写的
        System.out.println(result.apply("like"));//CAN'T FIND THIS KEY
    }
```

总结：学习了Function和Functions之后，不是很理解这样设计的原因，也不太清楚Function和Functions的实际用 处，Guava的参考资料上说：函数式编程在Guava核心库Collections里面有大量体现，学习到Collections的时候再深入理解吧。

分类：Guava学习 时间：2014-11-28 人气：1331

本文关键词： guava function Functions 对象转换

分享到：

相关文章

解析PHP将对象转换成数组的方法(兼容多维数组类型)	2013-10-21
Jquery的基本对象转换和文档加载用法实例	2014-06-30
Windows Powershell对象转换成文本	2015-04-01
Ruby中嵌套对象转换成json的方法	2014-03-05
JavaScript通过function定义对象并给对象添加toString()方法实例分析	2014-03-19
PowerShell中使用Out-String命令把对象转换成字符串输出的例子	2014-08-30
C#中的DataSet.string.DataTable.对象转换成Json的实现代码	2015-02-21
对象转换JSON格式	2012-11-30

js将类数组对象转换成数组对象

2015-02-24

iOS 开发

ios iphone ipad
objective-c sqlite safari
xcode phonegap cocoa
javascript macosx

Android 开发

android java eclipse
xml phonegap json
webview

Python 开发

python list django
flask tornado web.py
sqlalchemy virtualenv

JAVA 开发

java java-ee jar spring
hibernate struts tomcat

开发语言

java c c++ php perl
python javascript c#
ruby objective-c go
lua node.js erlang
scala bash actionscript

PHP 开发

php mysql apache
nginx 数组 mvc
codeigniter symfony
zend-framework

Ruby 开发

ruby ruby-on-rails
rubygems rvm macosx

搜索

搜索引擎 中文分词
全文检索 lucene solr

前端开发

html html5 css css3
javascript jquery json
ajax

数据库

数据库 mysql sqlite
oracle sql nosql redis
mongodb memcached

开发工具

vim emacs ide eclipse
xcode intellij-idea
textmate sublime-text
visual-studio git github
svn hg

开放平台

新浪微博 人人网 微信
腾讯微博 百度 街旁

Javascript 开发

javascript jquery yui
mootools node.js
chrome firefox firebug
internet-explorer

.NET 开发

.net c# asp.net
visual-studio mvc
microsoft

云计算

云计算 又拍云存储
七牛云存储
google-app-engine
sina-app-engine
amazon-web-services
百度云

服务器

maven eclipse sphinx analyzer facebook twitter

intellij-idea

linux unix ubuntu

windows-server centos

负载均衡 缓存 apache

nginx