



Java8的Predicate，让代码更简洁



ImportSource 发表于 ImportSource

694



11.11 智慧上云

云服务器企业新用户优先购，享双11同等价格

立即抢购

在我们的代码中，经常会编写是否为真的代码，比如用户名是否存在，客户是否存在等。类似如下代码：

```
public boolean exist(Long userId){  
  
    ...  
  
    return false;  
  
}
```

这样做已经很棒了。但你需要了解一个java8的Predicate。通过Predicate可以让你的代码更加的简洁。学习下Predicate吧。

Predicate是一个函数接口。它包含了一个接口方法和三个默认方法以及一个静态方法。

```
@FunctionalInterface  
public interface Predicate<T> {  
    boolean test(T t);  
    default Predicate<T> and(Predicate<? super T> other) {  
        Objects.requireNonNull(other);  
        return (t) -> test(t) && other.test(t);  
    }  
    default Predicate<T> negate() { return (t) -> !test(t); }  
    default Predicate<T> or(Predicate<? super T> other) {  
        Objects.requireNonNull(other);  
        return (t) -> test(t) || other.test(t);  
    }  
    static <T> Predicate<T> isEqual(Object targetRef) {  
        return (null == targetRef)  
            ? Objects::isNull  
            : object -> targetRef.equals(object);  
    }  
}
```

Predicate表示断定和假设的意思。

test

test接口就是让你实现判断的效果。最原始的就是去实现这个接口，然后写我们的判断逻辑，如下：

```
public class OriginPredicate {  
    public static void main(String[] args) {  
        Predicate<Integer> predicate=new Predicate<Integer>() {  
            @Override  
            public boolean test(Integer integer) {  
                return integer>10;  
            }  
        };  
    }  
}
```

你也看到有一部分灰色的代码，告诉我们这个可以被优化为lambda表达式，如下：

```
public class OriginPredicate {  
    public static void main(String[] args) {  
        Predicate<Integer> predicate= i -> i>10;  
    }  
}
```

你也可以把Predicate的实现单独抽离成一个实现类，方便重用。

```
public class CustomPredicate implements Predicate<Integer> {
    @Override
    public boolean test(Integer integer) {
        return integer.intValue() > 20;
    }
}
```

Predicate不仅可以单独在代码中使用，也可以在测试代码中用来做判断，同时还可以被用在Stream的filter中，用来做过滤。

```
public class ArrayStream {
    public static void main(String[] args) {
        Integer[] integers = new Integer[] {1, 2, 30, 20, 34, 25, 6, 9, 10};
        Arrays.stream(integers).filter(new CustomPredicate()).forEach(System.out::println);
    }
}
```

使用Predicate可以让你的判断逻辑代码更加的简洁和解耦，增加了可读性、可测试性，同时符合DRY原则。

DRY原则：(don't repeat yourself): writing code more than once is not a good fit for a lazy developer ;)It also makes your software more difficult to maintain because it becomes harder to make your business logic consistent。一句话：别写重复代码

现在你的代码看起来已经很炫酷了。但别急，接下来的几个操作会让你的代码看起来更加的清晰而明了。当你写了一个“大于20”的条件时，此时需求变了，需要一个新的条件 $20 < x < 30$ 。

此时你有两条路，一条路是直接修改Predicate的test方法中的逻辑。还有一条路是新建一个新的Predicate，然后和现有的组装成为一个新的Predicate。第一条路相信你经常走，接下来就来介绍下第二条路：组装。

and

首先新建一个LessThan30Predicate：

```
public class LessThan30Predicate implements Predicate<Integer> {
    @Override
    public boolean test(Integer integer) {
        return integer.intValue() < 30;
    }
}
```

然后和上面的CustomPredicate用and方法组装成为一个新的Predicate。如下：

```
public class ArrayStream {
    public static void main(String[] args) {
        Integer[] integers = new Integer[] {1, 2, 30, 20, 34, 25, 6, 9, 10};
        Arrays.stream(integers).filter(new CustomPredicate().and(new LessThan30Predicate())).forEach(System.out::println);
    }
}
```

输出结果：

25

or

同样也可以使用or来组装。

```
public class ArrayStream {
    public static void main(String[] args) {
        Integer[] integers = new Integer[] {1, 2, 30, 20, 34, 25, 6, 9, 10};
        Arrays.stream(integers).filter(new CustomPredicate().or(new LessThan30Predicate())).forEach(System.out::println);
    }
}
```

输出结果：

1
2
30
20
34
25
6
9
10

negate

你也可以使用negate方法把现有的Predicate变为否定的Predicate。

```
public class ArrayStream {
    public static void main(String[] args) {
        Integer[] integers = new Integer[] {1, 2, 30, 20, 34, 25, 6, 9, 10};
    }
}
```

```
Arrays.stream(integers).filter(new CustomPredicate().negate()).forEach(System.out::println);
    }
}
```

输出结果:

```
1
2
20
6
9
10
```

原文发布于微信公众号 - ImportSource (importsource)


原文发表时间: 2019-01-30

本文参与[腾讯云自媒体分享计划](#), 欢迎正在阅读的你加入, 一起分享。

发表于 2019-05-06

Java

举报



ImportSource
187 篇文章 · 54 人订阅

[订阅专栏](#)

- PPT是世界上最好的语言
- 面试题, 如何在千万级的数据中判断一个值是否存在?
- 图解Kafka消息是被怎么存储的?
- Java程序员和美国决裂的一天
- 自由软件永远是自由的, github你可以继续使用, 只要不是用于搞原子弹

我来说两句

0 条评论

[登录](#) 后参与评论

[上一篇: PPT是世界上最好的语言](#)

[下一篇: 每一个吹嘘自己996的老板都坏得很](#)

社区

[专栏文章](#)
[互动问答](#)
[技术沙龙](#)
[技术快讯](#)
[团队主页](#)
[开发者手册](#)
[智能钛AI](#)

活动

[原创分享计划](#)
[自媒体分享计划](#)

资源

[在线学习中心](#)
[技术周刊](#)
[社区标签](#)
[开发者实验室](#)

关于

[社区规范](#)
[免责声明](#)
[联系我们](#)

云社区



扫码关注云+社区
领取腾讯云代金券

