Java 关于Java8的Optional类的理解和使用整理

原创 Shaochen_ 于 2018-12-11 16:39:47 发布 1253 收藏 版权
分类专栏: 个人经验 文章标签: Java Optional

简介

- 什么是Optional

> Optional 类是一个可以为null的容器对象。如果值存在则isPresent()方法会返回true，调用get()方法会返回该对象。
> Optional 是个容器：它可以保存类型T的值，或者仅仅保存null。Optional提供很多有用的方法，这样我们就不用显式进行空值检测。
> Optional 类的引入很好的解决 空指针异常 。

- 为什么要使用Optional

> 在我们开发的过程中，有许多这样的场景，我们从某个方法中处理或者从数据库查询的时候，会得到一个null的结果集，比如去数据库查询名字叫'老司机'的用户，如果没有查到，数据库就会返回给我们一个null，但是我们有时候不想把null直接传递给前端，想默认传给前端'真英俊'，这样的操作我们就必须判断为空的操作，进行特殊处理，Optional给我们提供了很方便的方式，下面是我自己的理解和使用方式的整理，如果有错误，还请多多指教。

使用整理

1 将数据包装成Optional

> Optional类中提供了静态方法：Optional.of()；Optional.ofNullable();
> 首先来看一下他们的区别

```
public static void main(String[] args){
        String test = "";
        Optional<String> demo1 = Optional.of(test);
        System.out.println("of: " + demo1);
        Optional<String> demo2 = Optional.ofNullable(test);
        System.out.println("ofNullable" + demo2);
        // 下面是结果图片
    }
```

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java.exe" ...
Connected to the target VM, address: '127.0.0.1:51882', transport: 'socket'
Disconnected from the target VM, address: '127.0.0.1:51882', transport: 'socket'
of: Optional[]
ofNullableOptional[]

Process finished with exit code 0
```

> 并没有什么不一样的，那是因为我们包装的对象不是null，如果是null，使用of方法就会报空指针异常，而ofNullable就不会。

```
public static void main(String[] args){
        String test = null;
        Optional<String> demo2 = Optional.ofNullable(test);
```

```
        System.out.println("of: " + demo1);
        // 结果图在下面
    }
```

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java.exe" ...
Connected to the target VM, address: '127.0.0.1:51961', transport: 'socket'
ofNullableOptional.empty
Exception in thread "main" java.lang.NullPointerException
Disconnected from the target VM, address: '127.0.0.1:51961', transport: 'socket'
    at java.util.Objects.requireNonNull(Objects.java:203)
    at java.util.Optional.<init>(Optional.java:96)
    at java.util.Optional.of(Optional.java:108)
    at com.maxrocky.service.parking.ParkingServiceImpl.main(ParkingServiceImpl.java:84)

Process finished with exit code 1
```


> 但实际上ofNullable()只是对of()进行了一层包装，实际上是在里面进行了一层判断，如果是空的话，就会调用另一个empty()方法进行直接返回。

```
    /**
     * Returns an {@code Optional} describing the specified value, if non-null,
     * otherwise returns an empty {@code Optional}.
     *
     * @param <T> the class of the value
     * @param value the possibly-null value to describe
     * @return an {@code Optional} with a present value if the specified value
     * is non-null, otherwise an empty {@code Optional}
     */
    public static <T> Optional<T> ofNullable(T value) {
        return value == null ? empty() : of(value);
    }
```


2 获取Optional包装的值

> 获取Optional值的方法有很多，在这之中，包含了很多处理数据的方式，包括，如果是null的默认值，包括如果是null抛异常等。

2.1 public T orElse(T other)

> orElse方法需要一个参数，如果返回的数据是null，就会返回我们填入的那个参数

```
public static void main(String[] args){
        String test = "JiShaochen is a very cool boy !";
        String test2 = null;
        Optional<String> demo = Optional.of(test);
        Optional<String> demo2 = Optional.ofNullable(test2);
        String sc = demo.orElse("JiShaochen is not a very cool boy !");
        String sc2 = demo2.orElse("JiShaochen is not a very cool boy !");
        System.out.println("test is not null : " + sc);
        System.out.println("test is null : " + sc2);
        // 结果图如下:
}
```

```
Connected to the target VM, address: '127.0.0.1:52119', transport: 'socket'
Disconnected from the target VM, address: '127.0.0.1:52119', transport: 'socket'
test is not null : JiShaochen is a very cool boy !
test is null : JiShaochen is not a very cool boy !

Process finished with exit code 0
```

2.2 public T orElseThrow(Supplier<? extends X> exceptionSupplier) throws X

> 这个方法是在获取数据的时候，如果返回的数据是null，就会抛出异常进行处理。

```
1  public static void main(String[] args){
2          String test = "JiShaochen is a very cool boy !";
3          String test2 = null;
4          Optional<String> demo = Optional.of(test);
5          Optional<String> demo2 = Optional.ofNullable(test2);
6          String sc = demo.orElseThrow(RuntimeException::new);
7          System.out.println(sc);
8          String sc2 = demo2.orElseThrow(RuntimeException::new);
9          System.out.println(sc2);
10          // 结果图如下:
11     }
```

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java.exe" ...
Connected to the target VM, address: '127.0.0.1:52436', transport: 'socket'
Disconnected from the target VM, address: '127.0.0.1:52436', transport: 'socket'
JiShaochen is a very cool boy !
Exception in thread "main" java.lang.RuntimeException
    at java.util.Optional.orElseThrow(Optional.java:290)
    at com.maxrocky.service.parking.ParkingServiceImpl.main(ParkingServiceImpl.java:87)

Process finished with exit code 1
```


2.3 public boolean isPresent()

> 这个方法是判断Optional对象是否有内容，会返回一个布尔类型的值。

```
1      public static void main(String[] args){
2          String test = "JiShaochen is a very cool boy !";
3          boolean present = Optional.ofNullable(test).isPresent();
4          System.out.println(present);
5          // 结果图如下:
6      }
```

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java.exe" ...
Connected to the target VM, address: '127.0.0.1:52547', transport: 'socket'
Disconnected from the target VM, address: '127.0.0.1:52547', transport: 'socket'
true

Process finished with exit code 0
```

> 有了这个，我们不但可以对Optional对象进行判断，也可以使用orElse方法获取默认数据，可以说是非常方便了。

2.4 public Optional filter(Predicate<? super T> predicate)

> 如果没有，我们就会返回一个空的Optional对象。

```
1  public static void main(String[] args){
2          List<String> testList = new ArrayList<>();
3          testList.add("aa");
4          testList.add("ab");
5          testList.add("Abc");
6          testList.add("structure");
7
8          Optional<List<String>> demo = Optional.ofNullable(testList);
9          Optional<List<String>> result = demo.filter(s -> s.contains("aa"));
10         System.out.println(result);
11     }
```

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java.exe" ...
Connected to the target VM, address: '127.0.0.1:52627', transport: 'socket'
Disconnected from the target VM, address: '127.0.0.1:52627', transport: 'socket'
Optional[[aa, ab, Abc, structure]]

Process finished with exit code 0
```

2.5 public Optional map(Function<? super T, ? extends U> mapper)

> 甚至我们可以使用map对Optional对象进行处理，比如我们可以在里面把所有的字符串变成大写等操作。

```
1  public static void main(String[] args){
2          String test = "JiShaochen is a very cool boy !";
3          Optional<String> demo = Optional.of(test).map(s -> s.toUpperCase());
4          System.out.println(demo);
5      }
```

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java.exe" ...
Connected to the target VM, address: '127.0.0.1:52714', transport: 'socket'
Disconnected from the target VM, address: '127.0.0.1:52714', transport: 'socket'
Optional[JISHAOCHEN IS A VERY COOL BOY !]

Process finished with exit code 0
```

总结

> 总结差不多就这么多了，接下来如果有新的理解会继续更新的。