Java8特性之Optional：如何干掉空指针？

IT码徒 2023-05-30 23:11 Posted on 日本

Optional的作用是什么？他都有哪些方法？阿里规范点名说尽量用Optional来避免空指针，那么什么场景用Optional？本篇文章围绕这三点来进行讲解。

目录

- 一、Optional类的来源
- 二、Optional类是什么？
- 三、Optional类用法
- 四、代码示例
  - 1、创建Optional类
  - 2、判断Optional容器中是否包含对象
  - 3、获取Optional容器的对象
  - 4、过滤
  - 5、映射
- 五、什么场景用Optional？
  - 1、场景一
  - 2、场景二
  - 3、场景三
  - 4、场景四

一、Optional类的来源

到目前为止，臭名昭著的空指针异常是导致Java应用程序失败的最常见原因。以前，为了解决空指针异常，Google公司著名的Guava项目引入了Optional类，Guava通过使用检查空值的方式来防止代码污染，它鼓励程序员写更干净的代码。受到Google Guava的启发，Optional类已经成为Java 8类库的一部分。

二、Optional类是什么？

Optional 类(`java.util.Optional`) 是一个容器类，它可以保存类型T的值，代表这个值存在。或者仅仅保存null，表示这个值不存在。原来用 null 表示一个值不存在，现在 Optional 可以更好的表达这个概念。并且可以避免空指针异常。

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。

三、Optional类用法

Optional类的Javadoc描述如下：这是一个可以为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。
- 如果值不存在则 `isPresent()` 方法会返回false，调用 `get()` 方法会NPE。

创建Optional类对象的方法：

- `Optional.of(T t)` : 创建一个 Optional 实例，t必须非空；
- `Optional.empty()` : 创建一个空的 Optional 实例
- `Optional.ofNullable(T t)`：t可以为null

判断Optional容器中是否包含对象：

- `boolean isPresent()` : 判断是否包含对象
- `void ifPresent(Consumer<? super T> consumer)` ：如果有值，就执行Consumer接口的实现代码，并且该值会作为参数传给它。

获取Optional容器的对象：

- `T get()`: 如果调用对象包含值，返回该值，否则抛异常
- `T orElse(T other)` ：如果有值则将其返回，否则返回指定的other对象。
- `T orElseGet(Supplier<? extends T> other)` ：如果有值则将其返回，否则返回由Supplier接口实现提供的对象。

- `T orElseThrow(Supplier<? extends X> exceptionSupplier)` ：如果有值则将其返回，否则抛出由Supplier接口实现提供的异常。

过滤：

`Optional<T> filter(Predicate<? super <T> predicate)`：如果值存在，并且这个值匹配给定的 predicate，返回一个Optional用以描述这个值，否则返回一个空的Optional。

映射

- `<U>Optional<U> map(Function<? super T,? extends U> mapper)`：如果有值，则对其执行调用映射函数得到返回值。如果返回值不为 null，则创建包含映射返回值的Optional作为map方法返回值，否则返回空Optional。
- `<U> Optional<U> flatMap(Function<? super T, Optional<U>> mapper)`：如果值存在，就对该值执行提供的mapping函数调用，返回一个Optional类型的值，否则就返回一个空的Optional对象

四、代码示例

```
@Data
@AllArgsConstructor
@NoArgsConstructor
class Student {
    private String name;
    private Integer age;
}
```

1、创建Optional类

```
public void test1() {
 // 声明一个空Optional
 Optional<Object> empty = Optional.empty();

 // 依据一个非空值创建Optional
 Student student = new Student();
 Optional<Student> os1 = Optional.of(student);

 // 可接受null的Optional
 Student student1 = null;
 Optional<Student> os2 = Optional.ofNullable(student1);
}
```

2、判断Optional容器中是否包含对象

isPresent不带参数，判断是否为空，ifPresent可以选择带一个消费函数的实例。（isPresent和ifPresent一个是 is 一个是 if 注意一下哈）

```
public void test1() {
    Student student = new Student();
    Optional<Student> os1 = Optional.ofNullable(student);
    boolean present = os1.isPresent();
    System.out.println(present);

    // 利用Optional的ifPresent方法做出如下：当student不为空的时候将name赋值为张三
    Optional.ofNullable(student).ifPresent(p -> p.setName("张三"));
}
```

3、获取Optional容器的对象

```
public void test1() throws Exception {
    Student student = null;
    Optional<Student> os1 = Optional.ofNullable(student);
    // 使用get一定要注意，假如student对象为空，get是会报错的
    // java.util.NoSuchElementException: No value present
    Student student1 = os1.get();

    // 当student为空的时候,返回我们新建的这个对象,有点像三目运算的感觉
    Student student2 = os1.orElse(new Student("张三", 18));

    // orElseGet就是当student为空的时候，返回通过Supplier供应商函数创建的对象
    Student student3 = os1.orElseGet(() -> new Student("张三", 18));

    // orElseThrow就是当student为空的时候，可以抛出我们指定的异常
    os1.orElseThrow(() -> new Exception());
}
```

4、过滤

```
public void test1() {
    Student student = new Student("李四", 3);
    Optional<Student> os1 = Optional.ofNullable(student);
    os1.filter(p -> p.getName().equals("张三")).ifPresent(x -> System.out.println("OK"));
}
```

5、映射

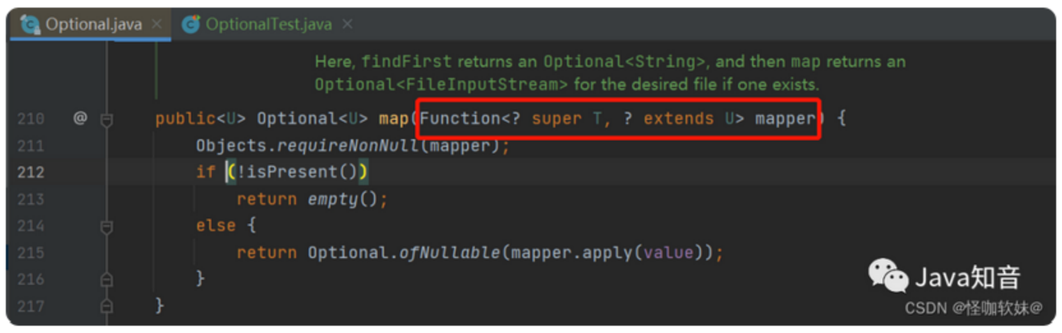
map代码示例：

```
public void test1() {
     Student student = new Student("李四", 3);
     Optional<Student> os1 = Optional.ofNullable(student);
     // 如果student对象不为空，就加一岁
     Optional<Student> emp = os1.map(e ->
     {
         e.setAge(e.getAge() + 1);
         return e;
     });
 }
```

这块的map说实话对lambda不是很熟练的 理解起来是很绕脑子的。

这里的map实际上就是用的Function函数，Function函数是有两个参数的，第一个是入参数据类型，第二个是返回数据类型。Function函数作用就是传入一个对象，然后返回一个对象，返回的对象类型可以自己设置。

- T 就是代表实例的泛型数据类型，就是谁调用的 入参 必须跟调用者泛型的数据类型一样。
- U 就是自己说了算，调用完map之后返回什么数据类型，那么U就设置什么



flatMap代码示例：flatMap跟map是一样的只不过他返回的是optional对象。

```
public static Optional<Integer> stringToInt(String s) {
     try {
         return Optional.of(Integer.parseInt(s));
     } catch (NumberFormatException e) {
         e.printStackTrace();
         return Optional.empty();
     }
 }
```

```
Optional.ofNullable(props.getProperty(name))
        .flatMap(OptionalUtils::stringToInt)
        .filter(i -> i>0)
        .orElse(0);
```

五、什么场景用Optional？

以前一直不懂Optional有啥用，感觉太无语了，Java8还把它当做一个噱头来宣传，最近终于发现它的用处了，当然不用函数式编程的话，是没感觉的；

如下提供了几个应用场景，基本上都是开发当中经常遇到的。

1、场景一

```
PatientInfo patientInfo = patientInfoDao.getPatientInfoById(consultOrder.getPatientId());
if (patientInfo != null) {
    consultInfoResp.setPatientHead(patientInfo.getHead());
}

// 使用Optional 和函数式编程，一行搞定，而且像说话一样
Optional.ofNullable(patientInfo).ifPresent(p -> consultInfoResp.setPatientHead(p.getHead()));
```

2、场景二

```
public void test1() throws Exception {
    Student student = new Student(null, 3);
    if (student == null || isEmpty(student.getName())) {
        throw new Exception();
    }
    String name = student.getName();
    // 业务省略...

    // 使用Optional改造
    Optional.ofNullable(student).filter(s -> !isEmpty(s.getName())).orElseThrow(() -> new Exception(
}

public static boolean isEmpty(CharSequence str) {
    return str == null || str.length() == 0;
}
```

3、场景三

```
public static String getChampionName(Competition comp) throws IllegalArgumentException {
    if (comp != null) {
        CompResult result = comp.getResult();
        if (result != null) {
            User champion = result.getChampion();
            if (champion != null) {
                return champion.getName();
            }
        }
    }
    throw new IllegalArgumentException("The value of param comp isn't available.");
}
```

这个在开发中是很常见的一种逻辑。去判读传进来的参数时候为空，或者是从数据库中获取的对象。由于某些原因，我们不能很流程的直接这样写。

```
comp.getResult().getChampion().getName()
```

上面的写法用Optional改写：

```
public static String getChampionName(Competition comp) throws IllegalArgumentException {
    return Optional.ofNullable(comp)
            .map(Competition::getResult)  // 相当于c -> c.getResult()，下同
            .map(CompResult::getChampion)
            .map(User::getName)
            .orElseThrow(()->new IllegalArgumentException("The value of param comp isn't available."
}
```

4、场景四

类型之间的转换，并且当没有值的时候返回一个默认值

```
int timeout = Optional.ofNullable(redisProperties.getTimeout())
       .map(x -> Long.valueOf(x.toMillis()).intValue())
       .orElse(10000);
```

来源：blog.csdn.net/weixin_43888891/article/details/124788806

— END —

【福利】2023 高薪课程，全面来袭（视频+笔记+源码）

PS：防止找不到本篇文章，可以收藏点赞，方便翻阅查找哦。

往期推荐

你见过哪些目瞪口呆的 Java 代码技巧？

加密后的敏感字段还能进行模糊查询吗？该如何实现？

面试官：如果要存ip地址，用什么数据类型比较好

卧槽！新来的妹纸rm -rf把公司整个数据库删没了，整个项目组慌了~

项目终于用上了xxl-job，真香！

一个注解实现 WebSocket 集群方案，这样玩才爽！

非常好用又炫酷的终端工具 -- Tabby

Read more