

# Optional 相關

Optional是Java SE 8中的一个新特性，它提供了一种优雅的方式处理可能空的值，避免了空指针异常的。

Optional类提供了以下常用方法：

- of()：创建一个包含指定非空值的Optional对象。如果传入null，会抛出NullPointerException异常。例如：

```
Optional<String> optional = Optional.of("hello");
```

- empty()：创建一个空的Optional对象。例如：

```
Optional<String> optional = Optional.empty();
```

- isPresent()：判断Optional对象是否包含非空值。例如：

```
Optional<String> optional = Optional.of("hello");
if (optional.isPresent()) {
    System.out.println("value is present: " + optional.get());
}
```

- get()：取Optional对象中的值，如果对象空抛出NoSuchElementException异常。例如：

```
Optional<String> optional = Optional.of("hello");
String value = optional.get();
System.out.println("value: " + value);
```

- orElse()：取Optional对象中的值，如果对象空返回指定的默认值。例如：

```
Optional<String> optional = Optional.empty();
String value = optional.orElse("default");
System.out.println("value: " + value);
```

- map()：对Optional对象中的值进行映射操作，返回一个新的Optional对象。例如：

```
Optional<String> optional = Optional.of("hello");
Optional<Integer> result = optional.map(s -> s.length());
System.out.println("result: " + result.get());
```

- filter()：对Optional对象中的值进行过滤操作，返回一个新的Optional对象。例如：

```
Optional<String> optional = Optional.of("hello");
Optional<String> result = optional.filter(s -> s.startsWith("h"));
System.out.println("result: " + result.get());
```

需要注意的是，Optional对象的操作通常采用链式调用的方式，如下所示：

```
Optional<String> optional = Optional.of("hello");
String result = optional.map(s -> s.toUpperCase())
    .orElse("default");
System.out.println("result: " + result);
```

在上面的示例中，我首先将字符串"hello"包装成Optional对象，然后使用map()方法将字符串转换成大写，最后使用orElse()方法取处理结果，如果对象空返回指定的默认值"default"。

optional.isPresent() 方法用于检查 Optional 对象是否包含非空值，如果对象中包含了一个非空的值，返回 true，否则返回 false。

如果使用 Optional.of() 方法创建一个 Optional 对象，传入了一个空引用，会抛出 NullPointerException 异常，而不是返回一个包含空值的 Optional 对象。因此，如果使用 Optional.of() 方法创建的对象中包含了一个空引用，调用 optional.isPresent() 方法会返回 false，而不是检查空字符串。

如果需要检查一个空字符串，可以使用 Optional.ofNullable() 方法创建一个 Optional 对象，例如：

```
String str = "";
```

```
Optional<String> optional = Optional.ofNullable(str);
if (optional.isPresent()) {
    System.out.println("str is not empty");
} else {
    System.out.println("str is empty");
}
```

在上面的示例中，我首先使用一个空字符串构建一个 `Optional` 对象，然后使用 `optional.isPresent()` 方法检查字符串是否空。由于字符串空，因此用 `optional.isPresent()` 方法会返回 `false`，所以最终输出结果 "str is empty"。

---

🕒 修訂版本 #2

★ 由 treeman 建立於 10 🕒 2023 10:04:30

✍ 由 treeman 更新於 5 🕒 2023 10:18:39