# jdk8

## optional

<https://www.cnblogs.com/zhangboyu/p/7580262.html>

### 使用方式

empty()，生成一个空optional对象

of(object value)，生成包含value值的optional对象

ofNullable(object value)，当value为null，调用empty()，否则同of()方法

get()，获取optional的value值

isPresent()，判断optional的value是否不为null

ifPresent(lambda表达式)，当optional的value不为null时，把value当做参数传入lambda表达式

orElse(object o)，当optional的value不为null时，返回该值，否则返回传递给它的参数值

orElseGet(lambda表达式)，有值的时候返回值，如果没有值，它会执行作为参数传入的 Supplier(供应者) 函数式接口，并将返回其执行结果

orElseThrow(() -> new IllegalArgumentException())，value为null时，抛自定义异常

map(lambda表达式)，把value当参数传入lambda表达式。当value为null，返回empty()，否则，把lambda表达式的返回值做参数，返回optional().ofNullable(value)

flatmap()，与map不同的是，value不为null，返回的直接是lambda表达式的返回值，其余与map逻辑相同。

filter(predicate lambda表达式)，为 true 的值。如果测试结果为 false，会返回一个空的 Optional。

### 注意

#### of和ofNullable

of()方法只能传入非null值，当value为null时，会抛空指针异常。

当传入值可能为null时，使用ofNullable，能保证value值不为null。当value为null，调用empty，返回一个null的optional。（ofNullable后使用get方法还是会返回null，同样会可能报空指针）

#### orElse和orElseGet

当value为null时，两个方法没有执行结果的差异。

但是value不为null时，orElse里的方法还是会执行，orElseGet里的lambda表达式则不会执行，这个会造成性能差异

### java9增强

#### or()

or(lambda表达式)，与orElse相似，不同的是，value为null时，or返回一个optional对象

#### ifPresentOrElse()

ifPresentOrElse() 方法需要两个参数：一个 Consumer 和一个 Runnable。如果对象包含值，会执行 Consumer 的动作，否则运行 Runnable。

如：Optional.ofNullable(user).ifPresentOrElse( u -> logger.info("User is:" + u.getEmail()),() -> logger.info("User not found"));

#### stream()

把实例转换为 Stream 对象，以调用stream的filter，map，collect等方法

### 使用

Jackson会把值为null的optional当做null，值不为null的把值当做对象属性值，而不是把optional当属性值。即，在对象序列化时，optional会被完全忽略掉。但是，不推荐把类的属性设置成optional，在转换对象某一条属性为optional时，使用map即可。

在构造函数中，没必要使用optional。

**optional主要在返回值中使用，防止多层次判断是否为null**

使用之前：

if (user != null) {

Address address = user.getAddress();

if (address != null) {

Country country = address.getCountry();

if (country != null) {

String isocode = country.getIsocode();

if (isocode != null) {

isocode = isocode.toUpperCase();

}

}

}

}

使用后：

String result = Optional.ofNullable(user)

.flatMap(u -> u.getAddress())

.flatMap(a -> a.getCountry())

.map(c -> c.getIsocode())

.orElse("default");

或：

String result = Optional.ofNullable(user)

.flatMap(User::getAddress)

.flatMap(Address::getCountry)

.map(Country::getIsocode)

.orElse("default");