Java 核心技术(高阶)

第八章 Java Stream

第四节 Optional类型

华东师范大学 陈良育

Stream的工作流程

- 流的工作流程
  - 创建一个流
  - 指定将流转换为其他流的中间操作，可包括多个步骤(惰性操作)
  - 应用终止操作，从而产生结果。这个操作会强制执行之前的惰性操作。这个步骤以后，流就再也不用了。

例子(1)

```
Integer[] a = new Integer[] {2,4,6,8};

Stream<Integer> s1 = Stream.of(a);
long countResult = s1.count();
System.out.println("the count result of s1 is " + countResult);

Stream<Integer> s2 = Stream.of(a);
countResult = s2.filter(n-> n>10).count();
System.out.println("the count result of s2 is " + countResult);
```

例子(2)

```
Integer[] a = new Integer[] {2,4,6,8};

Stream<Integer> s3 = Stream.of(a);
Optional<Integer> maxResult = s3.max((n1,n2)->n1-n2);
System.out.println("the max result of s3 is " + maxResult.get());

Stream<Integer> s4 = Stream.of(a);
maxResult = s4.filter(n-> n>10).max((n1,n2)->n1-n2);
System.out.println("the max result of s4 is " + maxResult.get());
```

```
Exception in thread "main" java.util.NoSuchElementException: No value present
        at java.util.Optional.get(Optional.java:135)
        at OptionalTest.main(OptionalTest.java:24)
```

NullPointerException

- 空指针异常
  - NullPointerException, 所有Java程序员的噩梦

```
String a = null;
System.out.println(a.length());

if(a != null)
{
    System.out.println(a.length());
}
```

```
Exception in thread "main" java.lang.NullPointerException
    at NullTest.main(NullTest.java:6)
```

Optional(1)

- Optional<T>
  - 一个包装器对象
  - 要么包装了类型T的对象，要么没有包装任何对象(还是null)
  - 如果T有值，那么直接返回T的对象
  - 如果T是null，那么可以返回一个替代物

Optional(2)

- Optional<T>

```
Optional<String> s1 = Optional.of(new String("abc"));
String s2 = s1.get();
System.out.println("s2: " + s2); //abc

Optional<String> s3 = Optional.empty();
String s4 = s3.orElse("def");
System.out.println("s4: " + s4); //def
```

Optional(3)

- Optional<T>创建
  - of方法
  - empty方法
  - ofNullable方法，对于对象有可能为null情况下，安全创建

```
Optional<String> s5 = Optional.of(new String("abc"));
Optional<String> s6 = Optional.empty();
String s7 = null;
Optional<String> s8 = Optional.ofNullable(s7);
//s7不为Null，s8就是s7，否则s8就为Optional.empty()
```

Optional(4)

- Optional<T>使用
  - get方法，获取值，不安全的用法
  - orElse方法，获取值，如果为null，采用替代物的值
  - orElseGet方法，获取值，如果为null，采用Lambda表达式值返回
  - orElseThrow方法，获取值，如果为null，抛出异常
  - ifPresent方法，判断是否为空，不为空返回true
  - isPresent(Consumer),判断是否为空，如果不为空，则进行后续Consumer操作,如果为空，则不做任何事情
  - map(Function),将值传递给Function函数进行计算。如果为空，则不计算

Optional(5)

- Optional<T>注意事项
  - 直接使用get，很容易引发NoSuchElementException异常

```
Exception in thread "main" java.util.NoSuchElementException: No value present
        at java.util.Optional.get(Optional.java:135)
        at OptionalTest.main(OptionalTest.java:24)
```

  - 使用isPresent判断值是否存在，这和判断null是一样的低效

```
if(u1 == null)
{
    u1 = new User("noname", 10);
}
```

```
if(u2.isPresent())
{
    System.out.println(u2.get().getName());
}
```

总结

- Optional
  - 数据对象容器
  - Java用来解决NullPointerException的一把钥匙
  - 在流运算中，避免对象是否null的判定

谢 谢！