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。

**类声明**

以下是一个 **java.util.Optional<T>** 类的声明：

public final class Optional<T> extends Object

**类方法**

|  |  |
| --- | --- |
| **序号** | **方法 & 描述** |
| 1 | **static <T> Optional<T> empty()**  返回空的 Optional 实例。 |
| 2 | **boolean equals(Object obj)**  判断其他对象是否等于 Optional。 |
| 3 | **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。 |
| 4 | **<U> Optional<U> flatMap(Function<? super T,Optional<U>> mapper)**  如果值存在，返回基于Optional包含的映射方法的值，否则返回一个空的Optional |
| 5 | **T get()**  如果在这个Optional中包含这个值，返回值，否则抛出异常：NoSuchElementException |
| 6 | **int hashCode()**  返回存在值的哈希码，如果值不存在 返回 0。 |
| 7 | **void ifPresent(Consumer<? super T> consumer)**  如果值存在则使用该值调用 consumer , 否则不做任何事情。 |
| 8 | **boolean isPresent()**  如果值存在则方法会返回true，否则返回 false。 |
| 9 | **<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。 |
| 10 | **static <T> Optional<T> of(T value)**  返回一个指定非null值的Optional。 |
| 11 | **static <T> Optional<T> ofNullable(T value)**  如果为非空，返回 Optional 描述的指定值，否则返回空的 Optional。 |
| 12 | **T orElse(T other)**  如果存在该值，返回值， 否则返回 other。 |
| 13 | **T orElseGet(Supplier<? extends T> other)**  如果存在该值，返回值， 否则触发 other，并返回 other 调用的结果。 |
| 14 | **<X extends Throwable> T orElseThrow(Supplier<? extends X> exceptionSupplier)**  如果存在该值，返回包含的值，否则抛出由 Supplier 继承的异常 |
| 15 | **String toString()**  返回一个Optional的非空字符串，用来调试 |

**注意：** 这些方法是从 **java.lang.Object** 类继承来的。

**Optional 实例**

我们可以通过以下实例来更好的了解 Optional 类的使用：

**Java8Tester.java 文件**

import java.util.Optional;

public class Java8Tester {

public static void main(String args[]){

Java8Tester java8Tester = new Java8Tester();

Integer value1 = null;

Integer value2 = new Integer(10);

// Optional.ofNullable - 允许传递为 null 参数

Optional<Integer> a = Optional.ofNullable(value1);

// Optional.of - 如果传递的参数是 null，抛出异常 NullPointerException

Optional<Integer> b = Optional.of(value2);

System.out.println(java8Tester.sum(a,b));

}

public Integer sum(Optional<Integer> a, Optional<Integer> b){

// Optional.isPresent - 判断值是否存在

System.out.println("第一个参数值存在: " + a.isPresent());

System.out.println("第二个参数值存在: " + b.isPresent());

// Optional.orElse - 如果值存在，返回它，否则返回默认值

Integer value1 = a.orElse(new Integer(0));

//Optional.get - 获取值，值需要存在

Integer value2 = b.get();

return value1 + value2; } }

执行以上脚本，输出结果为：

$ javac Java8Tester.java

$ java Java8Tester

第一个参数值存在: false

第二个参数值存在: true

10