Optional

# Java Optionals | Crash Course

<https://youtu.be/1xCxoOuDZuU>

Este es muy claro.

El proyecto amigoscode implementa todo lo que se dice en este tutorial.

<https://www.oracle.com/technical-resources/articles/java/java8-optional.html>

<https://www.baeldung.com/java-optional>

# La clase Optional<T>

Clase Optional se introdujo en Java 8. El propósito de la clase es proporcionar una solución para representar valores opcionales en lugar de referencias nulas. La idea es poder escribir APIs de tal modo que quien mire el código sepa inmediatamente cuándo un campo podría ser null. El beneficio de esto es poder usar programación funcional, y tener código limpio.

La programación funcional es un paradigma, tal como la programación orientada a objetos. No hay una descripción simple y completa, sino más bien una serie de distinciones y contrastes. Lo que sin duda sería erróneo, es suponer que estos dos paradigmas son mutuamente excluyentes. No lo son. Sí son dos cosas distintas. Son complementarios. Aclarado esto, vamos a usar un poco de programación funcional en este tutorial, y no vamos a entrar en más explicaciones.

# Cómo lidiar con NullPointerException

Una NullPointerException es un problema, porque es de tipo runtime. Es decir que, si se dispara, nuestra aplicación deja de funcionar. Esto es inaceptable.

Siempre se puede programar de modo defensivo, tratando de anticipar todas las instrucciones que podrían retornar null, para encerrarlas en bloques try catch.

También podemos ir controlando al retorno de cada función potencialmente problemática, y decidir un curso de acción basado en el valor retornado, null o not null.

En cualquiera de los dos casos fácilmente terminamos con un código complicado y feo.

La clase Optional<T> nos permite trabajar con sentencias que eventualmente podrían retornar null, sin el riesgo de caer en una null pointer exception, y sin vernos sometidos a la necesidad de programar de un modo ridículamente paranoico.

# Qué es un opcional

[https://docs.oracle.com/en/java/javase/17/docs/api//java.base/java/util/Optional.html](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Optional.html)

Un Optional es un objeto contenedor que puede contener un valor no nulo, pero también puede no contenerlo. No estamos diciendo que puede contener un valor nulo. Estamos que diciendo que puede no contener un valor no nulo. Es decir que el valor puede estar o no presente. Si está presente, entonces es no nulo. Y si no está presente entonces no es que sea nulo. Es que está no presente.

Y esto es lo que tiene de único y de útil. Siempre que se requiere obligatoriamente tener o retornar un objeto de un dado tipo, si aparece un valor nulo se produce una NullPointerException.

El Optional es una solución a este problema. Si hay un valor presente, isPresent() devuelve verdadero. Si no hay ningún valor presente, el objeto se considera vacío, y en ese caso isPresent() devuelve falso.

Se proporcionan métodos adicionales que dependen de la presencia o ausencia de un valor contenido, como orElse() (devuelve un valor predeterminado si no hay ningún valor presente) e ifPresent() (realiza una acción si hay un valor presente).

Esta es una clase basada en valores; los programadores deben tratar las instancias que son iguales como intercambiables y no deben usar instancias para la sincronización, o puede ocurrir un comportamiento impredecible. Por ejemplo, en una versión futura, la sincronización puede fallar.

**Nota API:**

Optional está diseñado principalmente para usarse como un tipo de retorno de método donde existe una clara necesidad de representar "sin resultado" y donde es probable que el uso de null cause errores. Una variable cuyo tipo es Optional nunca debe ser nula; siempre debe apuntar a una instancia de Optional.

# Métodos estáticos

La clase Optional tiene tres métodos estáticos. Es decir, son métodos de la clase, no de una instancia. La siguiente función prueba los tres métodos, y luego muestra los resultados:

    public static void staticMethods() {

        // Returns an empty Optional instance. No value is present for this Optional.

        showMe("Optional.empty()", Optional.empty());

        // Returns an Optional describing the given non-null value.

        showMe("Optional.of(456)", Optional.of(456));

        // Returns an Optional describing the given value, if non-null, otherwise

        // returns an empty Optional.

        showMe("Optional.ofNullable('Una cadena')", Optional.ofNullable("Una cadena"));

        showMe("Optional.ofNullable(null)", Optional.ofNullable(null));

    }

La siguiente función muestra los resultados llamando a los métodos pertinentes:

    public static void showMe(String title, Optional<?> optObj) {

        System.out.println("\n" + title + "\n");

        System.out.println("optObj: " + optObj);

        System.out.println("optObj.isPresent(): " + optObj.isPresent());

        System.out.println("optObj.isEmpty(): " + optObj.isEmpty());

    }

# Método orElse()