

(<http://baeldung.com>)

Java Opcional - orElse () vs orElseGet ()


Última modificación: 17 de mayo de 2018

por baeldung (<http://www.baeldung.com/author/baeldung/>)
(<http://www.baeldung.com/author/baeldung/>)

Java (<http://www.baeldung.com/category/java/>) +

Java 8 (<http://www.baeldung.com/tag/java-8/>)



Acabo de anunciar los nuevos módulos de **Spring 5** en 
REST With Spring:

>> COMPRUEBA EL CURSO →

1. Introducción

La API de *Opcional* normalmente tiene dos métodos que pueden causar confusión: *orElse ()* y *orElseGet ()*.

En este tutorial rápido, veremos la diferencia entre esos dos y exploraremos cuándo usar cada uno.

2. Firmas

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

Comencemos por lo básico con sus firmas:

```
1 public T orElse(T other)
2
3 public T orElseGet(Supplier<? extends T> other)
```

Claramente, *orElse()* toma cualquier parámetro de un tipo *T*, mientras que *orElseGet()* acepta una interfaz funcional de tipo *Proveedor* que devuelve un objeto de tipo *T*.

Ahora, en base a sus Javadocs

(<https://docs.oracle.com/javase/8/docs/api/java/util/Optional.html#orElse-T->):

- *orElse()*: devuelve el valor si está presente; de lo contrario, devuelve *otro*
- *orElseGet()*: devuelve el valor si está presente; de lo contrario, invoca a *otro* y devuelve el resultado de su invocación

3. Diferencias

Es fácil confundirse un poco con estas definiciones simplificadas, así que profundicemos un poco más y observemos algunos escenarios de uso reales.

3.1. *si no()*

Suponiendo que tenemos nuestro registrador

(<http://www.baeldung.com/java-logging-intro>), configurado correctamente, comencemos con la escritura de un simple código:

```
1 String name = Optional.of("baeldung")
2     .orElse(getRandomName());
```

Observe que *getRandomName()* es un método que devuelve un nombre aleatorio de una *lista* *<String>* de *nombres*:

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

```

1 public String getRandomName() {
2     LOG.info("getRandomName() method - start");
3
4     Random random = new Random();
5     int index = random.nextInt(5);
6
7     LOG.info("getRandomName() method - end");
8     return names.get(index);
9 }

```

Al ejecutar nuestro código, encontraremos a continuación los mensajes impresos en la consola:

```

1 getRandomName() method - start
2 getRandomName() method - end

```

El *nombre de la* variable contendrá *"baeldung"* al final de la ejecución del código.

Con él, podemos inferir fácilmente que el **parámetro de `orElse()` se evalúa incluso cuando se tiene un *Opcional* no vacío**.

3.2. `orElseGet()`

Ahora, intentemos escribir un código similar usando `orElseGet()`:

```

1 String name = Optional.of("baeldung")
2     .orElseGet(() -> getRandomName());

```

El código anterior no invocará el método `getRandomName()`.

Recuerde (del Javadoc) que el método `Supplier` pasado como argumento solo se ejecuta cuando un valor *opcional* no está presente.

Usar `orElseGet()` para nuestro caso, por lo tanto, es **mucho más cercano a su trabajo / función actual?**

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

4. Medición del impacto en el rendimiento

Ahora, para comprender también las diferencias en el rendimiento, usemos JMH (<http://www.baeldung.com/java-microbenchmark-harness>) y veamos algunos números reales:

```

1  @Benchmark
2  @BenchmarkMode(Mode.AverageTime)
3  public String orElseBenchmark() {
4      return Optional.of("baeldung").orElse(getRandomName());
5  }

```

Y *orElseGet ()*:

```

1  @Benchmark
2  @BenchmarkMode(Mode.AverageTime)
3  public String orElseGetBenchmark() {
4      return Optional.of("baeldung").orElseGet(() -> getRandomName());
5  }

```

Mientras ejecutamos nuestros métodos de referencia, obtenemos:

1	Benchmark	Mode	Cnt	Score	Error	Units
2	orElseBenchmark	avgt	20	60934.425 ± 15115.599		ns/op
3	orElseGetBenchmark	avgt	20	3.798 ± 0.030		ns/op

Como podemos ver, el impacto en el rendimiento puede ser sustancial incluso para un escenario de uso tan simple.

Sin embargo, los números anteriores pueden variar levemente, o ***ElseGet ()* claramente ha superado a *orElse ()* para nuestro ejemplo particular.**

Después de todo, *orElse ()* implica el cálculo del método *getRandomName ()* para cada ejecución.

5. ¿Qué es importante?

Además de los aspectos de rendimiento, otros factores que merecen la pena incluyen:

- ¿Y si el método ejecutara alguna lógica adicional? Por ejemplo, hacer algunas inserciones de DB o actualizaciones
- Incluso cuando asignamos un objeto al parámetro *orElse ()* :

```

1  String name = Optional.of("baeldung").orElse("Other");

```

todavía estamos creando el objeto "Otro" sin ninguna razón

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

Y es por eso que es importante para nosotros tomar una decisión cuidadosa entre `orElse ()` y `orElseGet ()` dependiendo de nuestras necesidades: **de forma predeterminada, tiene más sentido usar `orElseGet ()` cada vez, a menos que el objeto predeterminado ya esté construido y accesible directamente.** .

6. Conclusión

En este artículo, hemos aprendido los matices entre los *métodos Opcional* o `Else ()` y `OrElseGet ()` . También notamos cómo a veces esos conceptos simples pueden tener un significado más profundo.

Como siempre, el código fuente completo se puede encontrar en Github (<https://github.com/eugenp/tutorials/tree/master/core-java-8>) .

Acabo de anunciar los nuevos módulos de Spring 5 en REST With Spring:

>> VERIFIQUE LAS LECCIONES (/rest-with-spring-course#new-modules)

²  Deja una respuesta



Join the discussion...



1



1



0



☒ Suscribirse ▼

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior



2



Desarrollador principal

▲ el más nuevo

▲ más antiguo

▲ el más votado

Arquitecto

Gerente



Huésped

rswrc

"Todavía



estamos creando el objeto" Otro "sin ninguna razón," OK, pero el proveedor también es un objeto ".

+ 0 -

Responder

🕒 Hace 21 días



Huésped

incze



pero el proveedor está construido solo por necesidad. entonces, estarás mejor con

String name = Opcional.of ("baeldung"). OrElseGet (() -> "Otro")

+ 0 -

Responder

🕒 Hace 9 días

CATEGORÍAS

PRIMAVERA ([HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/](http://www.baeldung.com/category/spring/))

DESCANSO ([HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/](http://www.baeldung.com/category/rest/))

JAVA ([HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/](http://www.baeldung.com/category/java/))

SEGURIDAD ([HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/](http://www.baeldung.com/category/security-2/))

PERSISTENCIA ([HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/](http://www.baeldung.com/category/persistence/))

JACKSON ([HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/](http://www.baeldung.com/category/jackson/))

HTTPCLIENT ([HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/](http://www.baeldung.com/category/http/))

KOTLIN ([HTTP://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/](http://www.baeldung.com/category/kotlin/))

SERIE

TUTORIAL "VOLVER A LO BÁSICO" DE JAVA ([HTTP://WWW.BAELDUNG.COM/JAVA-TUTORIAL](http://www.baeldung.com/java-tutorial))

JACKSON JSON TUTORIAL ([HTTP://WWW.BAELDUNG.COM/JACKSON](http://www.baeldung.com/jackson))

TUTORIAL DE HTTPCLIENT 4 ([HTTP://WWW.BAELDUNG.COM/HTTPCLIENT-GUIDE](http://www.baeldung.com/httpclient-guide))

REST CON SPRING TUTORIAL ([HTTP://WWW.BAELDUNG.COM/REST-WITH-SPRING-SERIES/](http://www.baeldung.com/rest-with-spring-series/))

TUTORIAL DE SPRING PERSISTENCE ([HTTP://WWW.BAELDUNG.COM/PERSISTENCE-WITH-SPRING-SERIES/](http://www.baeldung.com/persistence-with-spring-series/))

SEGURIDAD CON SPRING ([HTTP://WWW.BAELDUNG.COM/SECURITY-SPRING](http://www.baeldung.com/security-spring))

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

ACERCA DE

ACERCA DE BAELDUNG ([HTTP://WWW.BAELDUNG.COM/ABOUT/](http://www.baeldung.com/about/))

LOS CURSOS ([HTTP://COURSES.BAELDUNG.COM](http://courses.baeldung.com))

TRABAJO DE CONSULTORÍA ([HTTP://WWW.BAELDUNG.COM/CONSULTING](http://www.baeldung.com/consulting))

META BAELDUNG ([HTTP://META.BAELDUNG.COM/](http://meta.baeldung.com/))

EL ARCHIVO COMPLETO ([HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE](http://www.baeldung.com/full_archive))

ESCRIBIR PARA BAELDUNG ([HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES](http://www.baeldung.com/contribution-guidelines))

CONTACTO ([HTTP://WWW.BAELDUNG.COM/CONTACT](http://www.baeldung.com/contact))

INFORMACIÓN DE LA COMPAÑÍA ([HTTP://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO](http://www.baeldung.com/baeldung-company-info))

TÉRMINOS DE SERVICIO ([HTTP://WWW.BAELDUNG.COM/TERMS-OF-SERVICE](http://www.baeldung.com/terms-of-service))

POLÍTICA DE PRIVACIDAD ([HTTP://WWW.BAELDUNG.COM/PRIVACY-POLICY](http://www.baeldung.com/privacy-policy))

EDITORES ([HTTP://WWW.BAELDUNG.COM/EDITORS](http://www.baeldung.com/editors))

KIT DE MEDIOS (PDF) ([HTTPS://S3.AMAZONAWS.COM/BAELDUNG.COM/BAELDUNG+-+MEDIA+KIT.PDF](https://s3.amazonaws.com/baeldung.com/baeldung+-+media+kit.pdf))

**¿Cuál de estos es el más
cercano a su trabajo /
función actual?**

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente