

Blog sobre Java EE

75€ de publicidad gratuita para usar en Google Ads.

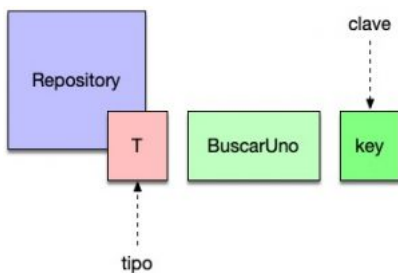
Continuar

Estás aquí: [Inicio](#) / [Arquitectura](#) / [Design-Patterns](#) / Java Optional Repository y JPA

Java Optional Repository y JPA

18 febrero, 2019 por [Cecilio Álvarez Caules](#) — 2 comentarios

El concepto de **Java Optional Repository** llega en Java 8 y hace referencia a cómo podemos usar los tipos Optional **dentro de nuestras clases clásicas de repositorio**. El ejemplo mas clásico es el método **BuscarUno** o **findOne()** a nivel de **Java Persistence API**. Este método nos permite buscar un elemento concreto por clave primaria utilizando el repositorio.



En muchas ocasiones el programa puede dar problemas en el caso de que no se encuentre ningún **elemento ya que cuando intentamos acceder a sus campos saltará una excepción de tipo NullPointerException**.

```
1 package repositorios.ejemplo1;
2
3 import javax.persistence.EntityManager;
4
5 import model.Cliente;
6
7 public class ClienteRepository {
8
9
10
11     private EntityManager entityManager;
12
13     public EntityManager getEntityManager() {
14         return entityManager;
15     }
16
17     public void setEntityManager(EntityManager entityManager) {
18         this.entityManager = entityManager;
19     }
20
21     public Cliente buscarUno(String dni) {
22
23         return entityManager.find(Cliente.class, "1");
24     }
25 }
26
27 package repositorios;
28
29 import javax.persistence.EntityManager;
```

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

ACEPTAR

plugin cookies

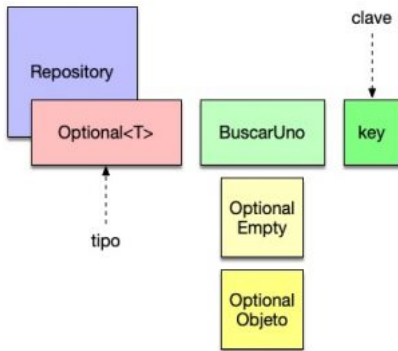
```

14 EntityManagerFactory emf = Persistence.createEntityManagerFactory("persistence-unit");
15 EntityManager em = emf.createEntityManager();
16
17 ClienteRepository repositorio = new ClienteRepository();
18
19 Cliente cliente=repositorio.buscarUno("1");
20 System.out.println(cliente.getNombre());
21
22
23 }
24
25 }

```

Java Optional Repository

Para evitar estos problemas podemos **apoyarnos en Java 8** y hacer uso de los tipos Optional que nos obligan a comprobar si un tipo esta a null antes de acceder a la información que almacena.



Veámoslos en código:

```

1 package repositorios.ejemplo2;
2
3 import java.util.Optional;
4
5 import javax.persistence.EntityManager;
6 import javax.persistence.EntityManagerFactory;
7 import javax.persistence.Persistence;
8
9 import model.Cliente;
10
11 public class ClienteRepository {
12     private EntityManager entityManager;
13
14     public EntityManager getEntityManager() {
15         return entityManager;
16     }
17
18     public void setEntityManager(EntityManager entityManager) {
19         this.entityManager = entityManager;
20     }
21
22
23
24
25     public ClienteRepository(EntityManager entityManager) {
26         super();
27         this.entityManager = entityManager;
28     }
29
30     public Optional<Cliente> buscarUno(String dni) {
31
32

```

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[ACEPTAR](#)[plugin cookies](#)

```
44     }  
45 }
```

Ahora la clase de repositorio hace uso de JPA pero apoyandose en tipos Optional de Java 8 de tal forma que cuando queramos acceder desde un cliente nos obligará a comprobar si existe un valor antes de imprimirlo por la pantalla.

```
1  package repositorios;  
2  
3  import java.util.Optional;  
4  
5  import javax.persistence.EntityManager;  
6  import javax.persistence.EntityManagerFactory;  
7  import javax.persistence.Persistence;  
8  
9  import model.Cliente;  
10 import repositorios.ejemplo2.ClienteRepository;  
11  
12  
13 public class Principal2 {  
14  
15     public static void main(String[] args) {  
16  
17         EntityManagerFactory emf = Persistence.createEntityManagerFactory("  
18             EntityManager em = emf.createEntityManager();  
19  
20         ClienteRepository repositorio = new ClienteRepository(em);  
21  
22         Optional<Cliente> cliente=repositorio.buscarUno("1");  
23  
24         if (cliente.isPresent()) {  
25             System.out.println(cliente.get().getNombre());  
26         }  
27  
28  
29  
30     }  
31  
32 }
```

Aprendamos a usar Java Optional Repository como un refactor importante a la hora de construir nuestras clases clásicas de repositorio.

1. [Java Optional Stream y reference methods](#)
2. [Java Optional ifPresent y como utilizarlo](#)
3. [Java 8 Optional y NullPointerExceptions](#)
4. [Java 8](#)



PDF

Archivado en: [Design-Patterns](#)

Comentarios

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[ACEPTAR](#)[plugin cookies](#)

En este caso al usar el Optional estamos añadiendo un `if (cliente.isPresent())` a la hora de imprimir el nombre. ¿Qué ventaja tiene esto con respecto a `if (null != cliente)`?

Saludos.

[Responder](#)



Juan Manuel Cabeza dice

18 febrero, 2019 en 12:54

Se gana bastante en legibilidad, es una expresión bastante más cercana al lenguaje natural.

[Responder](#)

Deja un comentario

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario

Nombre *

Correo electrónico *

Web

[PUBLICAR COMENTARIO](#)


Este sitio usa Akismet para reducir el spam. [Aprende cómo se procesan los datos de tus comentarios.](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.


ACEPTAR

plugin cookies

Java Herencia



Java JDBC



Servlets



Intro JPA




Mis Cursos de Java

Programación Orientada a Objeto en Java



Java APIS Core



Java Web



Pack Java Core



Arquitectura Java Solida con Spring



POPULAR

Spring Boot JPA y su configuración

Los Frameworks y su lado oscuro

Spring REST CORS y su configuración

Static Method vs instance method y su uso correcto

Java new String y la creación de objetos

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[ACEPTAR](#) [plugin cookies](#)

CONTACTO

contacto@arquitecturajava.com

LO MAS LEIDO

- [¿Qué es Spring Boot?](#)
- [Java Constructores this\(\) y super\(\)](#)
- [Java Optional Repository y JPA](#)
- [Usando Java Session en aplicaciones web](#)
- [El Principio de Substitución de Liskov](#)
- [Ejemplo de Java Singleton \(Patrones y ClassLoaders\)](#)
- [Introducción a Servicios REST](#)
- [Java Iterator vs ForEach](#)
- [Comparando java == vs equals](#)
- [Usando el patron factory](#)
- [Java Override y encapsulación](#)
- [REST JSON y Java](#)
- [Ejemplo de JPA , Introducción \(I\)](#)
- [Uso de Java Generics \(I\)](#)
- [¿Cuales son las certificaciones Java?](#)
- [¿Qué es Gradle?](#)
- [Eclipse refactor move y como utilizarlo](#)
- [¿Qué es un Microservicio?](#)
- [El modelo Entidad Relacion con DBDesigner](#)
- [El patrón de inyección de dependencia y su utilidad](#)
- [Mis Libros](#)
- [¿ Que es REST ?](#)
- [Spring MVC Configuración \(I\)](#)
- [Angular ngFor la directiva y sus opciones](#)
- [Eclipse JPA y clases de dominio](#)