

➤ Acceso a datos en Spring Framework

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Acceso a datos mediante JPA
- ✓ Clases Entidad y mapeo de datos con JPA

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

6.3

Start! 🏁

**Acceso a datos en
Spring Framework**

Clase Repositorio

CRUD con JPA

Clases de Repositorio
Recuperar, actualizar, eliminar un objeto en JPA

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



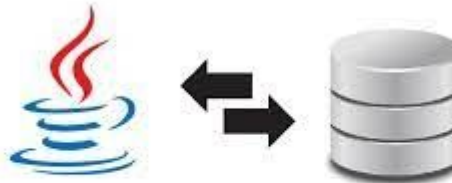
Implementar las clases Repositorio en la capa de acceso a datos mediante JPA



› Clases de Repositorio



Clases de Repositorio

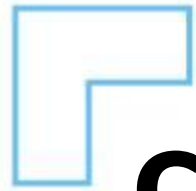


JAVA PERSISTENCE API



En el contexto de Spring y Java Persistence API (JPA), las clases de repositorio desempeñan un papel fundamental en la capa de persistencia de una aplicación. Estas clases facilitan la interacción entre la lógica de negocio y la capa de acceso a datos, proporcionando métodos convenientes para realizar operaciones CRUD en las entidades persistentes.





Clases de Repositorio



Las Clases de Repositorio en Spring Data JPA son interfaces o clases que extienden las interfaces proporcionadas por Spring Data y que heredan de **JpaRepository** o **CrudRepository**. Su principal objetivo es proporcionar una abstracción sobre las operaciones básicas de persistencia, permitiendo a los desarrolladores interactuar con la base de datos de manera más sencilla y orientada a objetos.





Clases de Repositorio

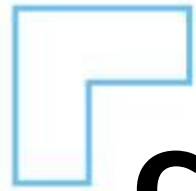


Para crear una interfaz repositorio, primero debemos anotarla con **@Repository**.
Para poder realizar esto es necesario tener la dependencia de **Spring Data JPA** en el archivo **pom.xml** de nuestra aplicación.



```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```





Clases de Repositorio



En este ejemplo, la interfaz **UsuarioRepository** extiende de **CrudRepository** y trabaja con la entidad **Usuario** cuya clave primaria es de tipo **String**.



```
8 @Repository
9 public interface UsuarioRepository extends CrudRepository<Usuario, String> {
10
11 }
```





Clases de Repositorio

Características Principales:



- **Generación de Consultas Automáticas:** Spring Data JPA realiza la generación automática de consultas basadas en el nombre del método. Por ejemplo, al definir un método `findById`, Spring Data JPA generará automáticamente la consulta SQL para buscar un usuario por su id.
- **Soporte para Métodos de Convención de Nombres:** Los métodos en las interfaces de repositorio pueden seguir una convención de nombres, y Spring Data JPA los interpretará automáticamente para crear consultas. Esto incluye métodos como `findBy`, `findAllBy`, `deleteBy`, etc.
- **Consultas Personalizadas:** Además de las consultas automáticas, los repositorios JPA permiten la definición de consultas personalizadas mediante el uso de anotaciones como `@Query`. Esto ofrece flexibilidad cuando se necesitan consultas más complejas.



Acceso a datos con JPA

Ventajas y Mejores Prácticas:

✕ **Abstracción y Simplicidad:** Las clases de repositorio abstraen gran parte de la complejidad del acceso a datos, permitiendo a los desarrolladores centrarse en la lógica de negocio.

Reducción de Código Boilerplate: Spring Data JPA reduce la necesidad de escribir código repetitivo y de bajo nivel para realizar operaciones CRUD.

Facilita las Pruebas Unitarias: Al proporcionar una interfaz, las clases de repositorio facilitan la creación de pruebas unitarias al permitir la sustitución de implementaciones para realizar pruebas de manera más efectiva.

Integración con Spring Boot: En aplicaciones basadas en Spring Boot, la configuración y la inicialización de repositorios son manejadas automáticamente, simplificando aún más el desarrollo.




➤ Recuperar, actualizar,
eliminar un objeto en JPA



Recuperar, actualizar, eliminar un objeto en JPA



Recuperar, actualizar y eliminar son operaciones fundamentales que permiten interactuar con la capa de persistencia de manera efectiva. A continuación, analicemos en detalle cómo se llevan a cabo estas operaciones en JPA con Spring, abordando aspectos teóricos, mejores prácticas y consideraciones relevantes.



Recuperar, actualizar, eliminar un objeto en JPA

CrudRepository es una interfaz genérica que recibe dos tipos. El primero es la clase que esta interfaz maneja y el segundo es el tipo de dato del ID de la entidad.

Simplemente con crear la **interfaz UsuarioRepository** y hacerla extender de **CrudRepository** ya podemos hacer uso de diversos métodos que se encuentran previamente declarados de manera automática (sin necesidad de escribir más código) .

```
8 @Repository
9 public interface UsuarioRepository extends CrudRepository<Usuario, String> {
10
11 }
```

Recuperar, actualizar, eliminar un objeto en JPA

Los métodos que nos provee CrudRepository son:

- **save**: guarda una entidad.
- **saveAll**: guarda las entidades de una lista iterable.
- **findById**: busca por el identificador.
- **existsById**: verifica si existe un identificador.
- **findAll**: devuelve todos los elementos para la entidad.
- **findAllById**: busca todos los elementos que tengan el identificador.
- **count**: devuelve el total de registros de la entidad.
- **deleteById**: elimina un registro según el identificador.
- **delete**: elimina la entidad.
- **deleteAllById**: elimina todos los elementos que correspondan con el identificador.
- **deleteAll(Iterable)**: elimina todos los elementos que se reciban en el parámetro.
- **deleteAll**: elimina todos los elementos.

Recuperar, actualizar, eliminar un objeto en JPA

```
@Repository
public interface UsuarioRepository extends CrudRepository<Usuario, String> {
    // Spring Data JPA proporciona métodos CRUD estándar
    // sin necesidad de escribir implementaciones.

    // Métodos para Leer (Retrieve) por ID:
    public Optional<Usuario> findById(Long id);
    public List<Usuario> findAll();

    // Métodos para Crear (Save):
    public Usuario save(Usuario usuario); // También se utiliza para actualizar

    // Métodos para Eliminar (Delete):
    public void deleteById(Long id);
    public void delete(Usuario usuario);
    public void deleteAll();
    public void deleteAll(Iterable<? extends Usuario> entities);
}
```

Recuperar, actualizar, eliminar un objeto en JPA

Si los métodos anteriores no son funcionales para realizar consultas específicas, podemos declarar métodos de búsqueda propios en la interfaz. Utilizando la convención de nomenclatura **jpql** podremos realizar el siguiente tipo de consultas:

```
13 @Repository
14 public interface UsuarioRepository extends CrudRepository<Usuario, String> {
15
16     @Query("SELECT u FROM Usuario u WHERE u.name = :name")
17     public Usuario buscarPorNombre(@Param("name") String name);
18 }
```

Recuperar, actualizar, eliminar un objeto en JPA

En esta query personalizada, podemos realizar una búsqueda a través de los parámetros señalados. La anotación `@Query` es necesaria para que Spring sepa que tiene que evaluarlo como una consulta a base de datos.

El parámetro `:"nombre"` hace referencia al señalado en la anotación `@Param("nombre")`, seguido por el tipo de dato a evaluar.

```
13 @Repository
14 public interface UsuarioRepository extends CrudRepository<Usuario, String> {
15
16     @Query("SELECT u FROM Usuario u WHERE u.nombre = :nombre")
17     public Usuario buscarPorNombre(@Param("nombre") String nombre);
18 }
```

Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro...

Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase.



LIVE CODING

Ejemplo en vivo

Completando la Wallet:

- *Vamos a crear las interfaces **repositorio** para Usuario y Cuenta*
- *Además, vamos a definir los métodos de búsqueda por mail para usuario.*

 **Tiempo: 15 minutos**



Ejercicio **CRUD con JPA**



CRUD con JPA



Contexto: 🙌

Vamos a continuar con el ejercicio de la clase anterior. En este caso vamos a agregar las interfaces de repositorio necesarias para gestionar los datos de un proyecto.



Consigna: 📝

Anotar una clase con @Repository una interfaz y hacerla extender de CrudRepository.

Anotar con @Query alguna consulta personalizada.

Tiempo 🕒: 15 minutos

○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ **Comprender los componentes y principales anotaciones de las clases de repositorio.**
- ✓ **Conocer la interfaz CrudRepository y los métodos que provee.**



#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 🙌 🙌 🙌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
 - a. *Lectura Modulo 6, Lección 3: páginas 16 - 19*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

Momento: ✚

Time-out!

🕒 5 min.

