AMS Bootcamp Fullstack Developer.

**JPA, Hibernate y Spring Data**



Índice

[1. Personal Notebook 4](#_Toc157633257)

[1.1 Datos alumno 4](#_Toc157633258)

[1.2 Personal Notebook 4](#_Toc157633259)

[2. Hibernate JPA y Spring Data 5](#_Toc157633260)

[2.1 Objetivos de aprendizaje 5](#_Toc157633261)

[2.2 ¿Qué es un ORM? 6](#_Toc157633262)

[2.2.1 Objetivos 6](#_Toc157633263)

[2.2.2 Recursos formativos 6](#_Toc157633264)

[2.2.3 Posibles preguntas 6](#_Toc157633265)

[2.2.4 Tareas 6](#_Toc157633266)

[2.2.5 Dudas 6](#_Toc157633267)

[2.3 Patrón DAO vs “Patrón” Repository 7](#_Toc157633268)

[2.3.1 Objetivos 7](#_Toc157633269)

[2.3.2 Recursos formativos 7](#_Toc157633270)

[2.3.3 Posibles preguntas 7](#_Toc157633271)

[2.3.4 Tareas 7](#_Toc157633272)

[2.3.5 Dudas 7](#_Toc157633273)

[2.4 Hibernate JPA 8](#_Toc157633274)

[2.4.1 Objetivos 8](#_Toc157633275)

[2.4.2 Mapeo de entidades y columnas. 8](#_Toc157633276)

[2.4.3 Mapeo de relaciones. 9](#_Toc157633277)

[2.4.4 Recursos formativos 9](#_Toc157633278)

[2.5 JPA Repository: Consultas 10](#_Toc157633279)

[2.5.1 Consultas a partir de propiedades 10](#_Toc157633280)

[2.5.2 Objetivos 11](#_Toc157633281)

[2.5.3 Recursos formativos 11](#_Toc157633282)

[2.5.4 Posibles preguntas 11](#_Toc157633283)

[2.5.5 Tareas 12](#_Toc157633284)

[2.5.6 Dudas 12](#_Toc157633285)

[2.6 JPA Repository: JPQL, Criteria 13](#_Toc157633286)

[2.6.1 ¿Qué es JPQL? 13](#_Toc157633287)

[2.6.2 Uso de @Query 13](#_Toc157633288)

[2.6.3 Objetivos 14](#_Toc157633289)

[2.6.4 Recursos formativos 14](#_Toc157633290)

[2.6.5 Posibles preguntas 14](#_Toc157633291)

[2.6.6 Tareas 14](#_Toc157633292)

[2.6.7 Dudas 14](#_Toc157633293)

[2.7 Ejercicio práctico: Consultas JPA 15](#_Toc157633294)

[2.7.1 Objetivos 15](#_Toc157633295)

[2.7.2 Recursos formativos 15](#_Toc157633296)

[2.7.3 Posibles preguntas 15](#_Toc157633297)

[2.7.4 Tareas 15](#_Toc157633298)

[2.7.5 Dudas 15](#_Toc157633299)

# Personal Notebook

## Datos alumno

* DAS
* Nombre Apellidos
* email
* URL de GitHub corporativo

## Personal Notebook

Este es tu cuaderno de trabajo. En él tendrás que registrar tu aprendizaje, tus avances y tus dudas. ¡Vas a generar tu propio material!

Puede ser una gran herramienta si la utilizas correctamente.

El cuaderno estará compartido con los tutores y el resto de colaboradores para ver tu progreso y ayudarte.

Al final del Bootcamp tendrás que entregarlo como evidencia de tu aprendizaje.



Instrucciones de uso:

El fichero está ubicado en tu carpeta del espacio que tenemos en Teams.

Siempre deberá ser editado en modo online y/o con guardado automático en la nube.

Responde a las preguntas con la información que necesites para dar una respuesta cualificada al formador o a tus compañeros. No es necesario copiar texto solo por “rellenar”. Cita las Fuentes en caso de utilizar otros recursos.

# Hibernate JPA y Spring Data

## Objetivos de aprendizaje

* Objetivo: **Adquirir los conocimientos necesarios para trabajar con sistemas ORM (Hibernate JPA) y con Spring Data.**
* Conceptos principales para adquirir:
  + Hibernate JPA,
  + Patrón DAO vs Repository,
  + Configuración,
  + Consultas y Relaciones.
  + Integración con Spring Data y MVC

Lecciones:

1. ¿Qué es un ORM?
2. Explicación de patrón DAO
3. Hibernate JPA: Entidades, Repositorios, Consultas
4. JPA Repository: Consultas (predefinidas,JPQL,Criteria…)
5. Ejercicio práctico: Consultas JPA
6. Spring Data JPA
7. Ejercicio completo Web + JPA

## ¿Qué es un ORM?

### Objetivos

1. **Introducción a ORM de manera genérica**
2. **Conocer los conceptos que hay detrás de estos enfoques**

### Recursos formativos

* + [(4) ¿Qué es un ORM? Object Relational Mapping - YouTube](https://www.youtube.com/shorts/Egr7EjiamYY))

### Posibles preguntas

* ¿Qué es un ORM? Indica los principales ORM.
* Si no uso un ORM ¿cómo puedo acceder a los datos desde Java?
* ¿Puedo utilizar cualquier librería ORM para mi proyecto Spring Boot?

### Tareas

- Investigar los pros y contras de utilizar ORM.

VIP: recuerda que tienes que responder a estas preguntas de forma que te ayude a explicárselas al formador o a los compañeros!!

### Dudas

¿?

¿?

¿?

¿?

¿?

VIP: registra las dudas que tengas para la sesión con el formador!!

¿?

## Patrón DAO vs “Patrón” Repository

### Objetivos

Entender la diferencia entre un patrón DAO y JPA Repository

### Recursos formativos

El patrón DAO es un patrón de diseño que proporciona una interfaz estándar para acceder a datos en bases de datos relacionales. El patrón DAO puede implementarse utilizando diferentes mecanismos para realizar consultas a la base de datos, como JDBC o un framework de ORM.

El patrón JPA Repository es un término que no se refiere directamente a un patrón de diseño, sino más bien a una característica proporcionada por la especificación JPA (Java Persistence API). El uso de un "Repository" en el contexto de JPA es común y se asocia con el concepto de un repositorio de datos que proporciona métodos para acceder y manipular entidades persistentes. Sin embargo, no es un patrón de diseño en sí mismo.

<https://www.baeldung.com/java-dao-vs-repository>

### Posibles preguntas

* ¿Qué es el patrón DAO? ¿Cuál es su propósito?
* Para que sirve la anotación @Repository.

### Tareas

NA

### Dudas

¿?

¿?

¿?

¿?

¿?

¿?

## Hibernate JPA

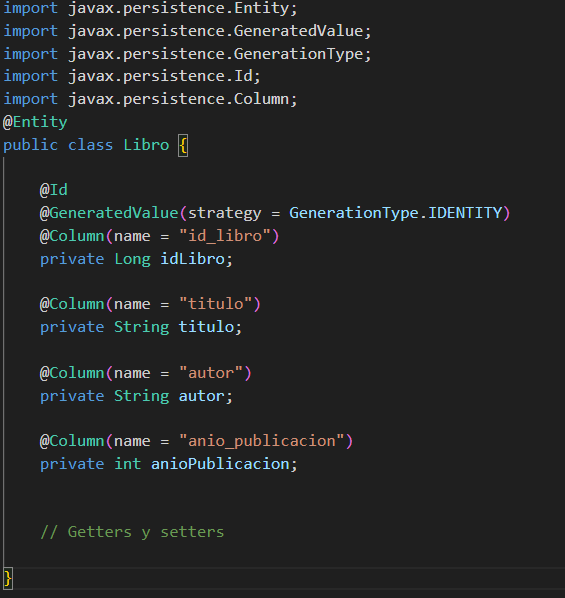
Hibernate es un framework de mapeo objeto-relacional (ORM) para Java. Fue una de las primeras implementaciones populares de la especificación JPA. Hibernate simplifica la interacción con bases de datos relacionales al mapear objetos de Java a tablas de bases de datos y viceversa, eliminando la necesidad de escribir consultas SQL directas.  
  
La librería que está incluida proyecto Spring Data JPA es Hibernate JPA, que es como por así decirlo, una versión más integrada con Spring Data del ORM, ya que Hibernate como tal existía desde antes.

### Objetivos

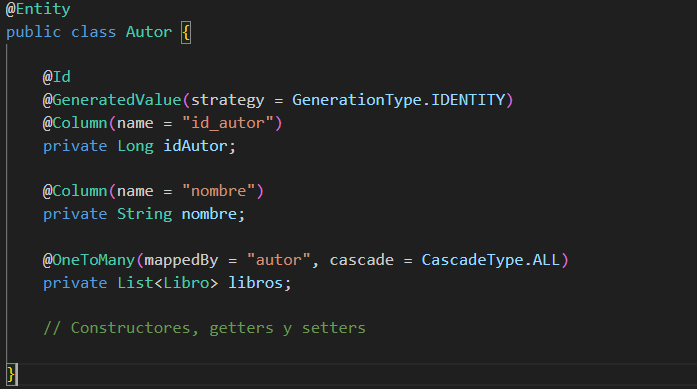
1. Aprender a mapear entidades y columnas.
2. Aprender a mapear los diferentes tipos de relaciones entre entidades.
3. Crear el esquema de la base de datos a partir del mapeo de Hibernate y no con scripts de bdd.
4. Utilizar secuencias autoincrementables en claves primarias.

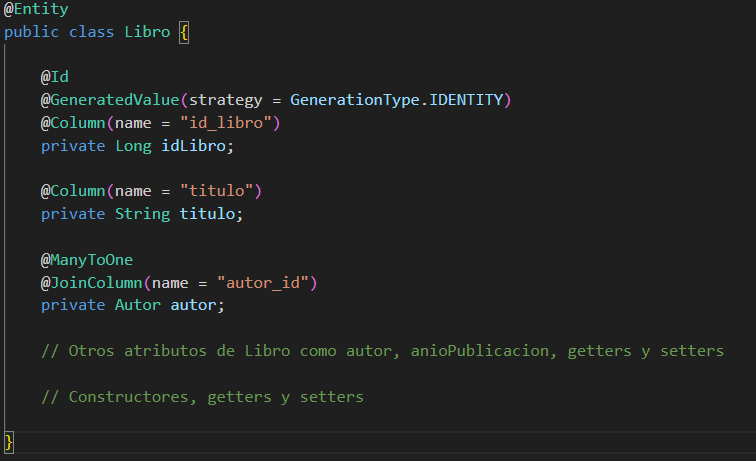
### Mapeo de entidades y columnas.

En este ejemplo se muestra el mapeo de una tabla Libro mediante el uso de anotaciones. Las anotaciones que usamos para mapear la tabla es @Entity (*Si el nombre de la clase coincide con el nombre de la tabla no es necesario añadir el atributo “name”*). También se usa la anotación @Column para definir las columnas en cada propiedad de nuestra clase. IMPORTANTE: Si nos equivocamos en algún nombre, o tipo no nos compilará el proyecto. Ya que Hibernate JPA se encarga de validar nuestro esquema.



### Mapeo de relaciones.

Esto sería otro ejemplo de como mapeariamos una relación entre dos entidades 1 a N.  
  




El atributo **mappedBy** se utiliza para indicar que la relación es propiedad de la entidad Libro, y **cascade = CascadeType.ALL** indica que las operaciones de persistencia (guardar, actualizar, eliminar) en un autor también deben afectar a sus libros.

### Recursos formativos

[Hibernate / JPA One-to-Many Mappings (howtodoinjava.com)](https://howtodoinjava.com/hibernate/hibernate-one-to-many-mapping/)

<https://www.youtube.com/watch?v=qocxxWNvHg4>

## JPA Repository: Consultas

Creación de interfaz con anotación @Repository, que extienda de JpaRepository o CrudRepository.

|  |
| --- |
| @Repository  public interface BookRepository extends JpaRepository<Book, Long> {  //Métodos adicionales según sea necesario  } |

Esta interfaz, al heredar de otra perteneciente a JPA, nos va a permitir hacer uso de un CRUD básico y operaciones usando el Id de la entidad. Los métodos más usados son **findById(), findAll(), save(), delete()...**

Es importante tener en cuenta que JpaRepository es una interfaz de JPA, pero existen algunas otras, como CrudRepository, que es la interfaz padre de JpaRepository, es decir, JpaRepository hereda los métodos de CrudRepository y además añade algunos más referentes a paginación etc.

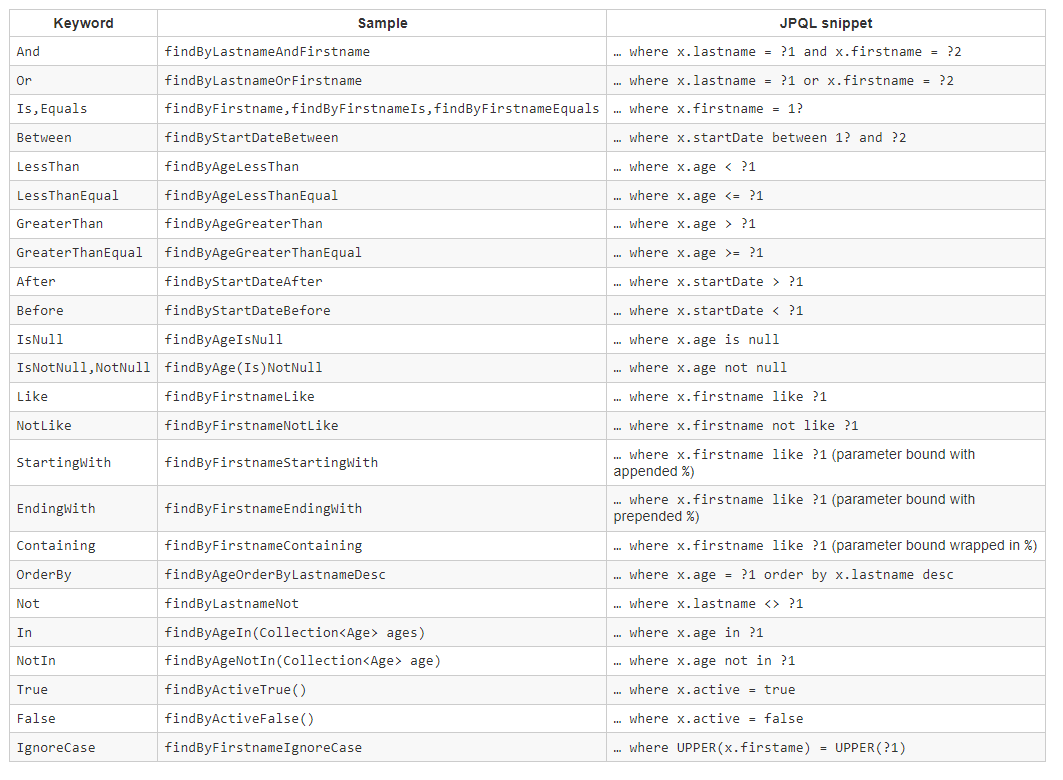
### Consultas a partir de propiedades

Si nosotros tenemos una entidad Libro con una propiedad título, que no es clave primaria, y queremos crear un método en el repositorio para buscar (Como si de una cláusula WHERE se tratase), solo tenemos que declarar un método de la siguiente forma.

|  |
| --- |
| @Repository  public interface BookRepository extends JpaRepository<Book, Long> {  List<Book> findByTittle(String tittle);  } |

JPA detecta automáticamente la propiedad nombre de la entidad libro en el propio nombre del método. Esto es un avance muy significativo ya que nos ahorra mucho tiempo con respecto a otro tipo de herramientas como JdbcTemplate o EntityManager.

Introduciremos las cláusulas necesarias en el propio nombre del método, y nos ahorraremos crear una implementación para definirlos, así de sencillo, a continuación se muestra una tabla con todos los JPA Criterias (Cláusulas) disponibles.



**Aplica lo mismo para los deleteBy.**

Por último, cabe destacar que, JpaRepository respeta las relaciones y las operaciones en cascada definidas en cada entidad. De manera que, si realizamos una consulta sobre una entidad padre, nos traerá la entidad o entidades hijas. Del mismo modo si realizamos operaciones de escritura o borrado, ésta las realizará en cascada, gracias a las anotaciones CascadeType de Hibernate.

### Objetivos

* **Comprender el funcionamiento de JPA, conocer los principales JPA Criterias.**
* **Cómo hacer consultas y las operaciones básicas de CRUD.**
* **Realizar consultas más complejas utilizando cláusula BY.**
* **Leer y escribir en cascada.**
* **Tomar conciencia de los efectos de determinados atributos de Hibernate.**

### Recursos formativos

* <https://www.youtube.com/watch?v=VlpIXkHU3EQ>

### Posibles preguntas

* ¿Qué significa JPA? ¿Para qué se usa?
* ¿Es lo mismo JPA que Hibernate?
* ¿Qué es una Entidad? ¿Cómo consigo que una clase sea una Entidad?
* Explica los tipos de relaciones de JPA y cuando usarlos.
* ¿Cuáles son los mecanismos de fetching? ¿diferencias? ¿implicaciones?
* ¿Qué métodos usamos para un CRUD? ¿Puedes explicar el ciclo de vida de una Entity?
* ¿Cuáles son los principales ficheros de configuración del Hibernate?

### Tareas

NA

### Dudas

* ¿Qué ocurre si necesito realizar subconsultas o consultas más complejas?
* ¿Debo crear una clase que implemente la interfaz @Repository?
* ¿Puedo desde la interfaz @Repository invocar a otro @Repository para traerme datos de otra entidad?

## JPA Repository: JPQL, Criteria

JpaRepository, tiene una limitación a la hora de realizar subconsultas, ya que no es capaz de realizar el mapeo a través de la declaración del método. Es por eso que se hace uso de JPQL y la anotación @Query.

### ¿Qué es JPQL?

JPQL no es mas que un sistema de mapeo de SQL a Java, es, básicamente intepretar una consulta SQL pero utilizando las clases Java (Entidades), en lugar de las tablas de la bbdd.  
  
De forma que si uno tiene una tabla TB\_LIBRO y un @Entity Libro puedo realizar la siguiente sentendia JPQL :  
  
**SELECT A FROM Libro A;**  
  
JPQL mapea automáticamente la palabra “Libro” con la entidad, y a diferencia de SQL en este caso no se utiliza nunca el “\*” asterisco.   
  
Siguiendo esta premisa, se puede hacer uso de las propiedades y no de las columnas.   
  
**SELECT A FROM Libro A WHERE A.nombre = ‘Título de libro’;**

Además de esto, JPQL nos permite parametrizar las consultas, haciendo uso de ( : ) delante del parámetro.  
  
**SELECT A FROM Libro A WHERE A.nombre = :nombreTitulo;**

### Uso de @Query

La anotación @Query se sitúa sobre la declaración del método en la interfaz @Repository, y esta anotación nos permite recibir una consulta JPQL. De esta forma podemos hacer uso de cláusulas JOIN para poder realizar subconsultas

|  |
| --- |
| public interface LibroRepository extends JpaRepository<Libro, Long> {  @Query("SELECT l.titulo FROM Libro l JOIN l.autores a WHERE a.nombre = :nombreAutor")  List<String> buscarTitulosPorNombreAutor(@Param("nombreAutor") String nombreAutor);  } |

### Objetivos

* **Comprender el funcionamiento de los mecanismos para realizar consultas en JPA e Hibernate**

### Recursos formativos

* <https://www.youtube.com/watch?v=FMkJhxopJcY>
* <https://www.baeldung.com/spring-data-criteria-queries>

### Posibles preguntas

* ¿Cómo puedo realizar consultas (select) con JPA/Hibernate?
* ¿Cuándo debo de usar uno u otro? ¿ventajas e inconvenientes? Explicación de su funcionamiento.
* ¿Qué parámetro tengo que usar para poder ver las consultas realizadas por consola?

### Tareas

NA

### Dudas

* ¿?
* ¿?
* ¿?
* ¿?
* ¿?
* ¿?
* ¿?

## Ejercicio práctico: Consultas JPA

### Objetivos

* + **Poner en práctica los conocimientos adquiridos mediante un caso de uso**

### Recursos formativos

NA

### Posibles preguntas

NA

### Tareas

* + El programa tiene que ser ejecutado desde el Main de la aplicación (mínimo) desde el propio IDE (la forma óptima será usando Junit).
  + La bbdd puede ser MySQL, Oracle o H2Database
  + Caso de uso:
  + Se quiere diseñar una base de datos relacional para gestionar los datos de los socios de un club náutico.
  + De cada socio se guardan los datos personales y los datos del barco o barcos que posee: número de matrícula, nombre, número del amarre y cuota que paga por el mismo.
  + Además, se quiere mantener información sobre las salidas realizadas por cada barco, como la fecha y hora de salida, el destino y los datos personales del patrón, que no tiene por qué ser el propietario del barco, ni es necesario que sea socio del club.
  + El programa tiene que realizar todas las operaciones CRUD necesarias para cada una de las entidades definidas.
  + Se deben anotar las entidades para utilizar los métodos de validación definidos por la especificación [JSR 380](https://jcp.org/en/jsr/detail?id=380).
  + Las operaciones de consulta tienen que realizarse utilizando los distintos métodos aprendidos durante la formación (consultas predefinidas, JPQL, Criteria) (se deja a cada alumno la decisión de definir las consultas que obtengan datos necesarios para la gestión del club náutico y los métodos JPA empleados)

### Dudas

¿?