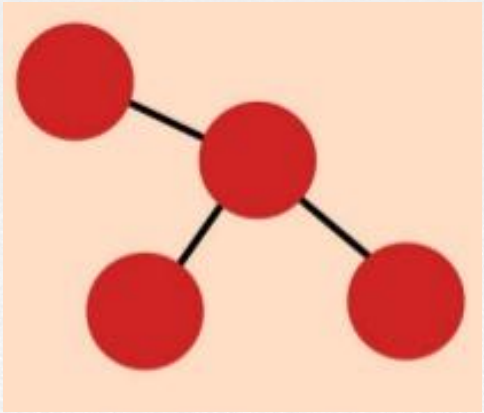
TIPS PARA CREAR UNA ENTIDAD EN HIBERNATE

INTRODUCCIÓN

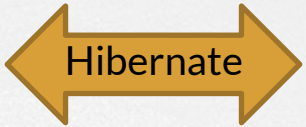
OpenWebinars

HIBERNATE

- ORM (object relational mapping)



Objetos Java



Tablas en la base de datos relacional

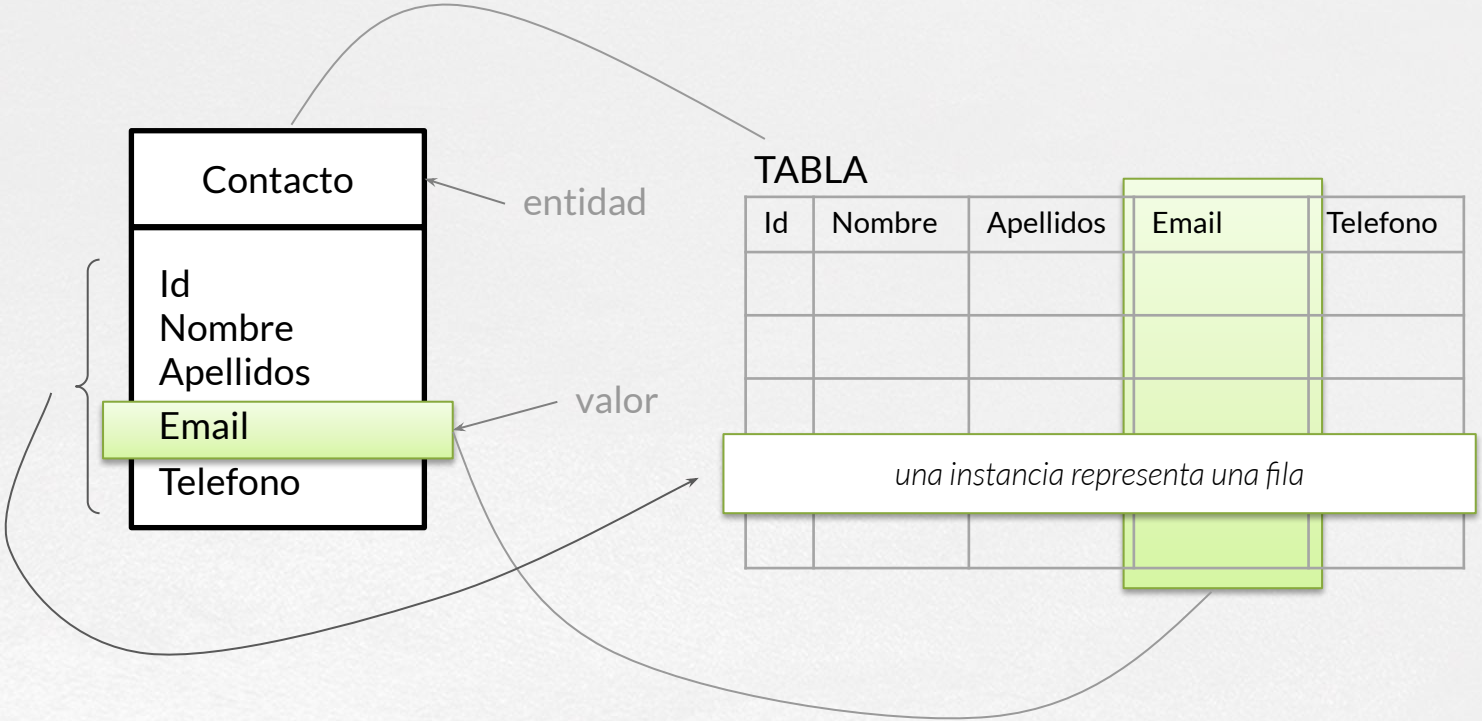
JPA (Jakarta Persistence API)

- Antes conocido como Java Persistence API
- Estándar que define cómo debe ser la persistencia de objetos Java en una base de datos relacional.
- Varias implementaciones
  - Hibernate
  - EclipseLink
  - ...

ENTIDADES

- Una entidad no es más que un objeto java (POJO)
- Se trata de la representación de la información persistida en una tabla de la base de datos.
- Una instancia de una entidad representa una fila de una tabla en base de datos.

ENTIDADES


Contacto
Id
Nombre
Apellidos
Email
Telefono
entidad
valor
TABLA
Id
Nombre
Apellidos
Email
Telefono
una instancia representa una fila

BASE DE DATOS H2

- Base de datos relacional
- Contenida en un único fichero *jar*: servidor, cliente, tools, …
- Almacenamiento: en memoria, fichero, …

PROYECTO BASE

- Proyecto Maven
- Dependencias Hibernate (+ JPA), H2
- Plugin para ejecutar
- Algún código de base para centrarnos solamente en las entidades.

TIPS PARA CREAR UNA ENTIDAD EN HIBERNATE

CARACTERÍSTICAS DE UNA ENTIDAD EN JPA

OpenWebinars

ENTIDADES EN JPA

- Clase Java (POJO)
- Anotada con *@Entity*
- Que incluya un *@Id*
- Que tenga un constructor sin argumentos (*public*, *protected*)
- Que no sea *enum*, *interface* (¿tampoco *record*?)
- *y todavía falta algún requisito o característica más*

ENTIDADES EN JPA

- La clase no debe ser final (ni métodos ni atributos*)
- Debe ser una clase top-level (no una clase interna)
- Puede ser concreta o abstracta
- Puede heredar de una clase que no sea una entidad
- *y algo más …*

ENTIDADES EN JPA

- **Si una instancia de entidad se va a utilizar de forma remota como un objeto separado, debe implementar la interfaz *Serializable*.**
- Es decir, si nuestra entidad se va a utilizar de alguna manera fuera de la JVM, debería implementar esta interfaz.
- En la práctica, no muchas veces sucede.

ENTIDADES EN JPA

- ***El estado de una entidad viene representado por el estado de sus propiedades. Estas deben ser accedidas directamente sólo por métodos de la clase.***
- En la práctica, que no olvidemos añadir los *getters/setters* para los atributos persistentes.

TIPS PARA CREAR UNA ENTIDAD EN HIBERNATE

CARACTERÍSTICAS DE UNA ENTIDAD EN HIBERNATE (NATIVO)

OpenWebinars

ENTIDADES EN HIBERNATE

Características menos restrictivas que en JPA

- Constructor sin argumentos (*public*, *protected*, *package*)
- No tiene porque ser una clase top-level
- Las clases o métodos getters/setters pueden ser finales, pero en general no es buena idea.
- Se pueden exponer variables de instancia como *public*, aunque esto no es buena práctica.

TIPS PARA CREAR UNA ENTIDAD EN HIBERNATE

HERRAMIENTAS AUXILIARES PARA LA CREACIÓN DE ENTIDADES

OpenWebinars

PROJECT LOMBOK

- Herramienta que tunea nuestro IDE (Eclipse, IDEA, …)
- Ya no tendremos que escribir más un método getter, setter, toString
- Basado en el uso de anotaciones

PROJECT LOMBOK

- @Getter, @Setter
- @NoArgsConstructor, @AllArgsConstructor
- @EqualsAndHashCode, @ToString

Anotaciones derivadas

- @Data (r/w), @Value (r/o)

ENTIDADES CON LOMBOK

- Constructor sin argumentos, getters y setters
- @NoArgsConstructor, @Getter, @Setter
- Hay quien usa @Data (¿esto es buena práctica?)
- Se puede añadir @Builder
- Tanto para JPA como para Hibernate nativo

TIPS PARA CREAR UNA ENTIDAD EN HIBERNATE

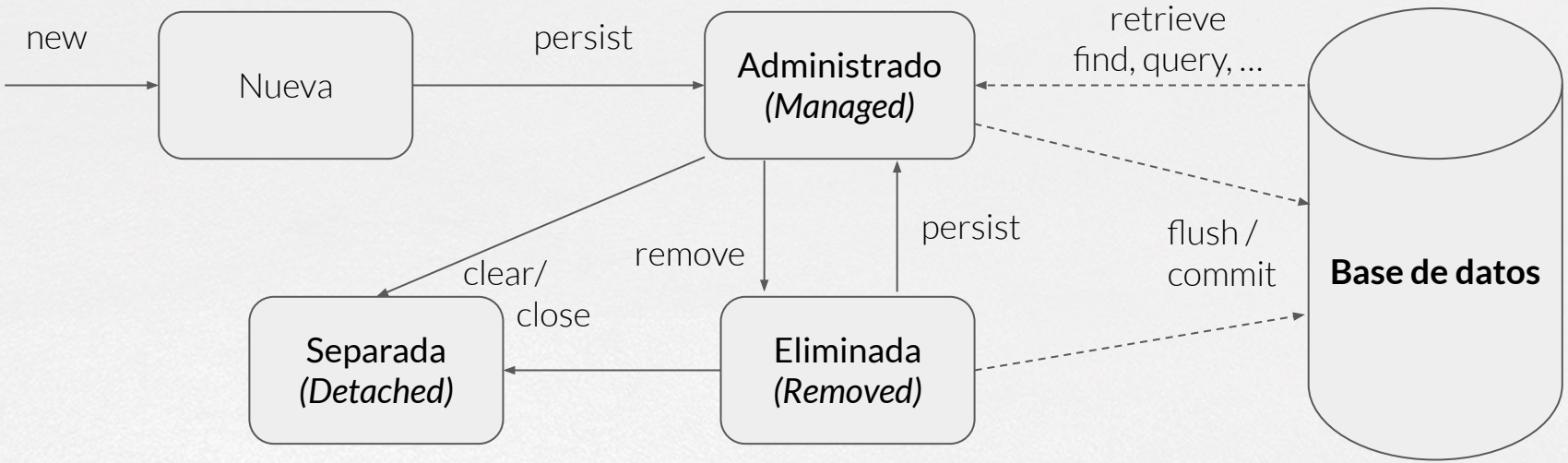
ADAPTACIÓN DE LOS MÉTODOS EQUALS Y HASHCODE

OpenWebinars

EQUALS, HASHCODE y TOSTRING

- No, no nos habíamos olvidado de ellos.
- La documentación oficial de JPA/Hibernate no dice ni que sean obligatorios, ni que dejen de serlo.
- Sí que nos da algunas pistas sobre cómo implementarlos.
- **Con todo, es un tema que provoca controversia entre los expertos en este campo.**

PERO ANTES... ESTADOS DE UNA ENTIDAD



OBLIGACIÓN DE IMPLEMENTAR EQUALS Y HASHCODE

- Solamente es obligatorio en un caso
- Si estamos utilizando como identificador una clase (propia).
- Normalmente aplicado a identificadores compuestos (por más de un atributo).
- En este caso, dicha clase debe implementar *equals()* y *hashCode()*.

OBLIGACIÓN DE IMPLEMENTAR EQUALS Y HASHCODE

- ¿Y el resto de casos?
- La documentación oficial nos recomienda NO IMPLEMENTARLOS.
- Esto hace que se delegue en las implementaciones de *java.lang.Object*.

PERO, ¿DÓNDE ESTÁ EL PROBLEMA?

- Habitualmente, dos instancias de una clase son iguales cuando todos sus atributos lo son.
- En el caso de las entidades, que representan una fila de una tabla con una clave primaria, esto no es así:
    - Dos filas son iguales cuando los valores de su clave primaria lo son
    - **Por tanto, dos entidades deberían ser iguales si son iguales los valores de su ID**

PERO, ¿DÓNDE ESTÁ EL PROBLEMA?

- Con Hibernate/JPA podemos tener más de una Session / EntityManager abiertos a la vez.
- El estado puntual de las instancias de la clase puede que sea diferente.

PERO, ¿DÓNDE ESTÁ EL PROBLEMA?

```java
@Entity
@Getter @Setter
@NoArgsConstructor @AllArgsConstructor
public class Book {

    3 usages
    @Id @GeneratedValue
    private Long id;

    private String title;

    private String author;
```

```java
@Entity
@Getter @Setter
@AllArgsConstructor @NoArgsConstructor
public  class Library {

    @Id @GeneratedValue
    private Long id;

    private String name;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "book_id")
    private Set<Book> books = new HashSet<>();

}
```

PERO, ¿DÓNDE ESTÁ EL PROBLEMA?

```
Book book1 = doInJPA(this::entityManagerFactory, entityManager -> {
        return entityManager.find(Book.class, 1L);
});

Book book2 = doInJPA(this::entityManagerFactory, entityManager -> {
        return entityManager.find(Book.class, 1L);
});

assertFalse(book1 == book2);
```

```
doInJPA(this::entityManagerFactory, entityManager -> {
        Set<Book> books = new HashSet<>();

        books.add(book1);
        books.add(book2);

        assertEquals(2, books.size());
});
```

ENTONCES ¿EQUALS BASADO EN ID?

```java
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Book book = (Book) o;
    return Objects.equals(id, book.id);
}

@Override
public int hashCode() {
    return Objects.hash(id);
}
```

ENTONCES ¿EQUALS BASADO EN ID?

```
public static void testBooks(EntityManager entityManager) {

    Library l = new Library();
    l.setName("Openwebinars Library");
    entityManager.getTransaction().begin();
    entityManager.persist(l);
    entityManager.getTransaction().commit();

    Book book1 = new Book();
    book1.setTitle("El Quijote");
```

```
Book book2 = new Book();
book2.setTitle("Hibernate for dummies");

l.getBooks().add(book1);
l.getBooks().add(book2);

entityManager.getTransaction().begin();
entityManager.persist(l);
entityManager.getTransaction().commit();
```

ENTONCES ¿EQUALS BASADO EN ID?

- Esta solución no siempre es factible.

```
l.getBooks().add(book1);
l.getBooks().add(book2);

entityManager.getTransaction().begin();
entityManager.persist(book1);
entityManager.persist(book2);
entityManager.persist(l);
entityManager.getTransaction().commit();
```

¿Y SI NO? @NATURALID

- Anotación de Hibernate (no estándar JPA)
- Indica que, si bien no es el ID, es un atributo propio de la clase diferente en todas las instancias.
- Posiblemente la mejor aproximación
- ¡PROBLEMA! No siempre está disponible.

¿Y SI NO? @NATURALID

```
3 usages
@NaturalId
private String isbn;

Luis Miguel López Magaña *
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Book book = (Book) o;
    return Objects.equals(isbn, book.isbn);
}

Luis Miguel López Magaña *
@Override
public int hashCode() {
    return Objects.hash(isbn);
}
```

¿Y SI NO? @NATURALID

```
Library l = new Library();
l.setName("Openwebinars Library");
entityManager.getTransaction().begin();
entityManager.persist(l);
entityManager.getTransaction().commit();

Book book1 = new Book();
book1.setTitle("El Quijote");
book1.setIsbn("978-123456789");

Book book2 = new Book();
book2.setTitle("Hibernate for dummies");
book2.setIsbn("978-987654321");
```

```
l.getBooks().add(book1);
l.getBooks().add(book2);

entityManager.getTransaction().begin();
entityManager.persist(book1);
entityManager.persist(book2);
entityManager.persist(l);
entityManager.getTransaction().commit();
```

EN CONCLUSIÓN

- Si tenemos un *natural-id*, usarlo para implementar equals y hashCode()
- Si no lo tenemos, y no usamos id autogenerados, usarlos para implementar estos métodos.
- Si no lo tenemos, y usamos id-autogenerados
  - en hashCode podemos devolver una constante
  - implementar equals basado en el ID