JPA

Caso de uso

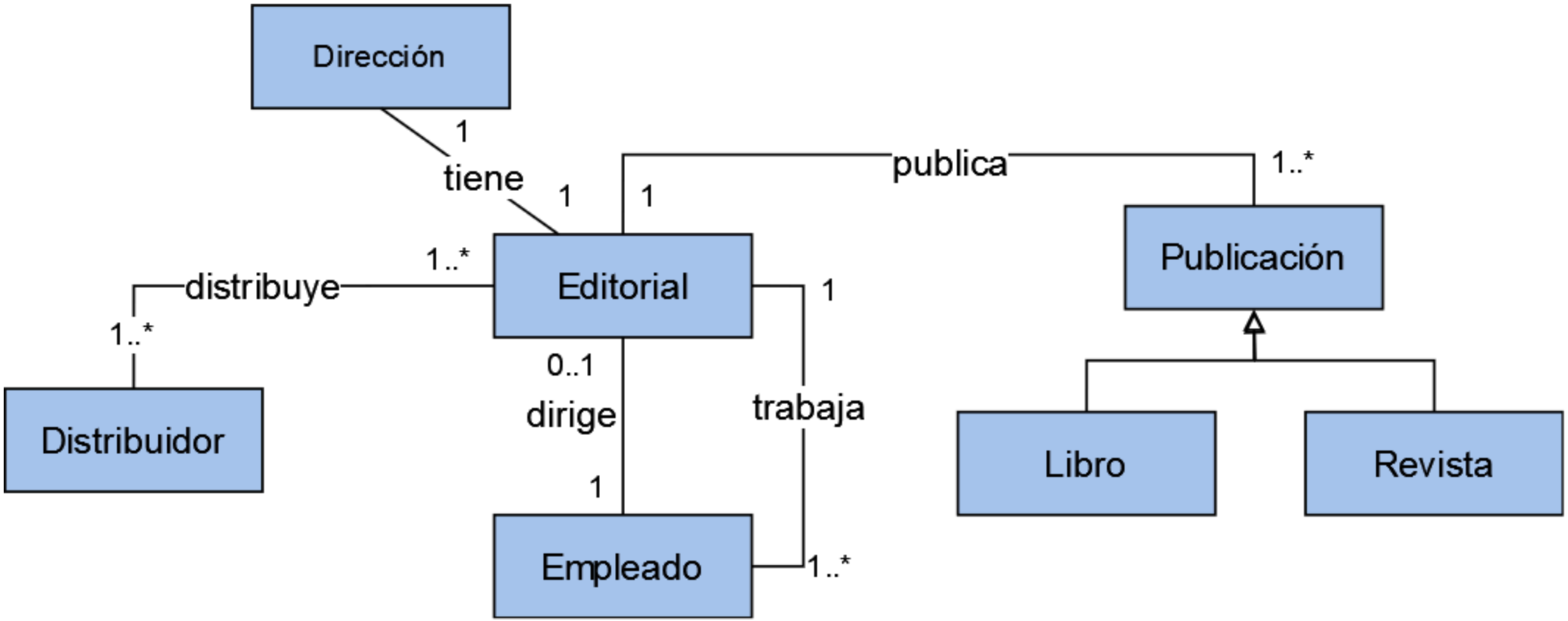
Mapeo de entidades y asociaciones

Aplicaciones Distribuidas

Curso 2025/2026

Introducción

- Para trabajar conceptos de JPA vamos a modelar las siguientes entidades y sus relaciones:



Dependencias

```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.28</version>
    </dependency>
    <dependency>
        <groupId>org.eclipse.persistence</groupId>
        <artifactId>eclipselink</artifactId>
        <version>2.7.15</version>
    </dependency>
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>javax.persistence-api</artifactId>
        <version>2.2</version>
    </dependency>
</dependencies>
```

Fichero `persistence.xml`

- Se crea dentro una carpeta `META-INF` en `src`.
- Define una o varias unidades de persistencia. Por cada unidad de persistencia:
  - Indica las clases que se van a mapear
  - Indica las porpiedades de conexión a la base de datos
  - Indica opciones de conexión del proveedor de JPA

```
<persistence-unit name="editoriales">
    <class>modelo.Editorial</class>
    <properties>
        <property name="javax.persistence.jdbc.driver"
            value="com.mysql.cj.jdbc.Driver" />
        <property name="javax.persistence.jdbc.user" value="root" />
        <property name="javax.persistence.jdbc.password"
            value="practicas" />
        <property name="javax.persistence.jdbc.url"
            value="jdbc:mysql://localhost:3306/sector_editorial?serverTimezone=CET" />
            ...
```

Clase EntityManagerHelper

- Nos ayuda a gestionar los objetos `EntityManager`.
- Crear una factoria de EntityManager a partir del nombre de la unidad de persistencia indicado en el fichero `persistence.xml`.
- Contiene todos los métodos para crear y destruir objetos `EntityManager` en un entorno multihilo.

```
public class EntityManagerHelper {
    private static EntityManagerFactory entityManagerFactory;
    private static final ThreadLocal<EntityManager> entityManagerHolder;

    static {
        entityManagerFactory = Persistence.createEntityManagerFactory("editoriales");
        entityManagerHolder = new ThreadLocal<EntityManager>();
    }
    public static EntityManager getEntityManager() {
       //...
```

Creación del esquema de bases de datos

- EclipseLink autogenera las tablas en la base de datos para todas las entidades anotadas

```
<property name="eclipselink.ddl-generation"
                value="create-or-extend-tables" />
```

- La base de datos idicada en la cadena de conexión debe existir previamente.

```
<property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/sector_editorial?serverTimezone=CET" />
```

- En **MySQL**:

```
CREATE SCHEMA sector_editorial;
```

Mapeo de entidad

```java
@Entity
@Table(name="editorial")
public class Editorial implements Serializable{
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name="id")
    private String id;
    @Column(name="nombre")
    private String nombre;
    @Column(name="fecha_fundacion")
    private LocalDate fechaFundacion;
    @Enumerated(EnumType.STRING)
    @Column(name="genero")
    private Genero genero;
    @Transient
    private long anyos;
    @Embedded
    private Direccion direccion;
    @ElementCollection
    @CollectionTable(name="telefono")
    private List<String> telefonos;
    //...
}
```

Asociaciones entre Editorial y Empleado

Un empleado trabaja en una editorial y la editorial puede tener varios empleados.

Para la asociación **Empleado trabaja en Editorial**, en `Empleado` añadimos:

```
@ManyToOne()
@JoinColumn(name="editorial")
private Editorial editorial;
```

La hacemos bidireccional añadiendo en `Editorial`:

```
@OneToMany(mappedBy = "editorial")
private List<Empleado> empleados;
```

Importante indicar `mappedBy` para que JPA sepa que ambos atributos pertenecen a la misma asociación.

Asociaciones entre Editorial y Empleado

Una editorial tiene un director, que será uno de sus empleados y cada empleado es director de una o de ninguna editorial.

En `Editorial` incluimos:

```
@OneToOne
private Empleado director;
```

¿Como hacemos la asociación bidireccional?

Crear Empleado - Director en una Editorial existente

```
private static Integer addEmpleadoDirector(String editorialId) {
    Empleado empleadoDirector = new Empleado();
    empleadoDirector.setNombre("Paolo");
    //..

    EntityManager em = EntityManagerHelper.getEntityManager();
    try {
        em.getTransaction().begin();

        Editorial editorial = em.find(Editorial.class, editorialId);
            // el empleado trabaja en la editorial
        empleadoDirector.setEditorial(editorial);

        em.persist(empleadoDirector);

        editorial.setDirector(empleadoDirector);

        em.getTransaction().commit();

    } catch (Exception ex) {
        //..
}
```

Asociaciones entre Editorial y Distribuidor

Relación muchos a muchos. En `Editorial` incluimos:

En `Editorial` incluimos:

```
@ManyToMany
   @JoinTable(name = "editorial_distribuidor", joinColumns = {
           @JoinColumn(name = "editorial_fk") },
           inverseJoinColumns = { @JoinColumn(name = "distribuidor_fk") })
   private ArrayList<Distribuidor> distribuidores;
```

Puede ser unidireccional o bidireccional añadiendo en `Distribuidor`:

```
@ManyToMany(mappedBy = "distribuidores")
private ArrayList<Editorial> editoriales;
```

Añadir Distribuidor a Editorial

```
Distribuidor distribuidor = new Distribuidor();
distribuidor.setCIF("2w34556789fdaaafsdf7643226633");
distribuidor.setNombre("Distribuciones Mercurio");

EntityManager em = EntityManagerHelper.getEntityManager();

try {
    em.getTransaction().begin();
    Editorial editorial = em.find(Editorial.class, editorialId);
    editorial.addDistribuidor(distribuidor);
    em.getTransaction().commit();

} catch (Exception ex) {
    //...
```

- Tal cual está no va a funcionar porque no hemos persistido distribuidor.

Añadir Distribuidor a Editorial

Podemos añadir el `persist` que falta:

```
em.persist(distribuidor);
    //...
```

O podemos modificar la asociación en `Editorial` para que se haga persistencia en cascada

```
@ManyToMany(cascade = CascadeType.PERSIST)
    @JoinTable(name = "editorial_distribuidor", joinColumns = {
            @JoinColumn(name = "editorial_fk") },
            inverseJoinColumns = { @JoinColumn(name = "distribuidor_fk") })
    private ArrayList<Distribuidor> distribuidores;
```