3 Fundamentos de JPA: Mapeo de entidades

3.3 - Fundamentos de JPA: Mapeo de entidades

Ficheros de mapeo XML → **Anotaciones JPA**

1. **Eliminar** el archivo **Videojuego.hbm.xml** (y la carpeta mappings)
2. **Modificar** la línea de mapping del fichero **hibernate.cfg.xml**

```
<mapping resource="mappings/Videojuego.hbm.xml"/>
```

↓

```
<mapping class="model.Videojuego" />
```

3. **Modificar** la clase **Videojuego.java añadiendo** las **anotaciones** que veremos a continuación

3.3 – Fundamentos de JPA: Mapeo de entidades

Anotaciones JPA

Anotaciones de clase

- **@Entity:** marca la clase como una entidad JPA
- **@Table:** asocia la clase a una tabla
    - **name = “Videojuego”:** parámetro opcional que indica el nombre concreto de la tabla. Si no se especifica, JPA utilizará el nombre de la clase como nombre de tabla

3.3 – Fundamentos de JPA: Mapeo de entidades

Anotaciones JPA

Anotaciones de atributos

- **@Id:** marca el atributo como clave primaria
- **@GeneratedValue(strategy = GenerationType.IDENTITY):** para claves primarias autogeneradas
- **@Column:** para definir las columnas de la tabla
  - **name = “”:** define el nombre de la columna. En caso de no definirse utilizará el nombre del atributo
  - **nullable = true/false:** si el campo es NOT NULL o no
  - **length = X:** define el tamaño máximo en campos de tipo texto
  - **precision = X:** define el número de dígitos enteros de un BigDecimal
  - **scale = X:** define el número de decimales en un BigDecimal
  - **unique = true/false:** si la columna es única en base de datos
- **@Temporal:** especifica el tipo de dato temporal:
  - **TemporalType.DATE:** indica que solo queremos almacenar la fecha
  - **TemporalType.TIME:** almacena solo la hora
  - **TemporalType.TIMESTAMP:** almacena la fecha y hora completas
- **@Lob:** permite almacenar grandes cantidades de texto (String) o datos binarios (byte[]) como imágenes o videos
- **@Transient:** indica a JPA que ignore el campo y no lo persista en la base de datos. Útil para propiedades calculadas o valores temporales

3.3 – Fundamentos de JPA: Mapeo de entidades

Anotaciones JPA

Videojuego.java

Si volvéis a ejecutar el código no os creará nada porque la tabla y los campos ya existen (no actualiza las propiedades)

```java
@Entity
@Table(name = "Videojuego")
public class Videojuego {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int videojuegoID;

    @Column(name = "Titulo", nullable = false, length = 100, unique = true)
    private String titulo;

    @Column(name = "Plataforma", nullable = false, length = 20)
    private String plataforma;

    @Column(name = "Anio_Lanzamiento", nullable = false)
    private int anioLanzamiento;

    @Column(name = "Precio", precision = 10, scale = 2, nullable = false)
    private BigDecimal precio;

    @Temporal(TemporalType.DATE)
    @Column(name = "Fecha_Actualizacion")
    private Date fechaActualizacion;

    @Column(name = "Disponible", nullable = false)
    private boolean disponible;

    public Videojuego() {}

    // Getters y setters
```

3.3 – Fundamentos de JPA: Mapeo de entidades

Anotaciones JPA

Videojuego.java

Investigad y probad las anotaciones de validación de Bean Validation de Hibernate Validator como @Min y @Max para liminar los años de publicación entre los años 1900 y 2100