DOCUMENTO DE EXPLICACIÓN

Práctica Hibernate – Sistema de Gestión de Biblioteca

Índice

---

1. ¿Qué hemos hecho?

Hemos creado un **sistema de gestión de biblioteca digital** que permite:

✅Registrar información básica

- **Libros** (título, ISBN, editorial)
- **Autores** (nombre)
- **Bibliotecas** (nombre, dirección)
- **Usuarios** (DNI, nombre, contacto)

✅Gestionar préstamos

- Registrar préstamos de libros
- Controlar disponibilidad de ejemplares
- Registrar devoluciones

✅Generar informes

- Ver qué libros tiene cada biblioteca
- Consultar libros prestados actualmente
- Listar libros con sus autores

---

2. ¿Qué es Hibernate?

Hibernate es un **"traductor"** entre:

- **Java** (nuestro programa)
- **Base de datos** (donde guardamos la información)

Sin Hibernate

```
// Tendríamos que escribir SQL a mano
String sql = "INSERT INTO libro (titulo, isbn) VALUES ('1984', 'ISBN-111')";
// Y luego procesar resultados manualmente
```

Con Hibernate

```
// Solo trabajamos con objetos Java
Libro libro = new Libro("1984", "ISBN-111");
session.persist(libro);  // Hibernate escribe el SQL automáticamente
```

Ventajas

- No necesitamos saber SQL avanzado
- Menos código y menos errores
- Código más limpio y fácil de mantener

---

3. Partes del proyecto

3.1 Las entidades (`domain/`)

Son como **plantillas** que representan las tablas de la base de datos.

```
@Entity  // Hibernate sabe que esta clase es una tabla
public class Libro {
    @Id  // Clave primaria
    @GeneratedValue  // Se genera automáticamente
    private Long id;

    private String titulo;
    private String isbn;
}
```

Entidades del proyecto

- **Autor** – Los escritores
- **Libro** – Las obras (ideas)
- **Ejemplar** – Las copias físicas
- **Biblioteca** – Donde están los libros
- **Persona** – Los usuarios
- **Prestec** – Los préstamos registrados

3.2 El Manager (`dao/Manager.java`)

Es el **administrador del sistema**, donde está la lógica principal.

```
public static Prestec addPrestec(Exemplar exemplar, Persona persona, ...) {
    // 1. Comprobar disponibilidad
    // 2. Crear préstamo
    // 3. Marcar ejemplar como no disponible
}
```

Funciones principales

- `addLibro()` – Registrar un libro
- `addPrestec()` – Realizar un préstamo
- `registrarRetornPrestec()` – Devolver un libro
- **Consultas HQL** – Generar informes

3.3 Las pruebas (`Main.java`)

Sirven para comprobar que todo funciona correctamente.

```
// Crear autores y libros
Autor autor1 = Manager.addAutor("George Orwell");
Libro libro1 = Manager.addLibro("1984", "ISBN-111");

// Realizar un préstamo
Prestec prestec = Manager.addPrestec(ejemplar, usuario, fecha);
```

---

4. Explicación de las relaciones

Diferencia clave: Libro vs Ejemplar

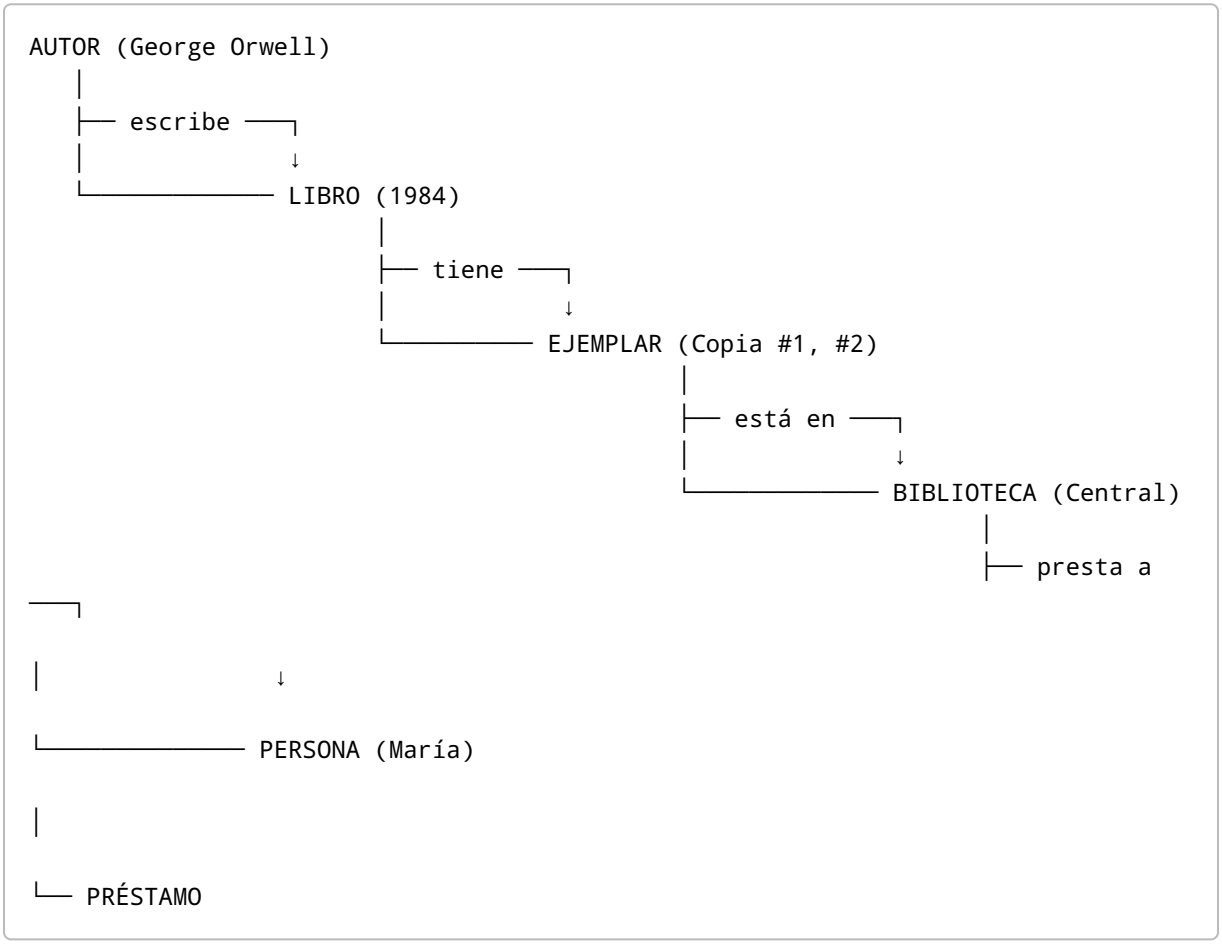
Libro	Ejemplar
La obra, la idea	La copia física
"1984" como concepto	Libro concreto con código
Puede tener varios autores	Está en una biblioteca
No se presta	Sí se presta

Ejemplo:

- Existe **1 libro** llamado *1984*
- La biblioteca puede tener **3 ejemplares** físicos de ese libro

---

Relaciones entre entidades



Tipos de relaciones

- **Uno a Muchos (OneToMany)**
- Un libro tiene muchos ejemplares
- Una biblioteca tiene muchos ejemplares
- **Muchos a Muchos (ManyToMany)**
- Un autor escribe muchos libros
- Un libro tiene muchos autores
- **Muchos a Uno (ManyToOne)**
- Un préstamo pertenece a una persona
- Un préstamo pertenece a un ejemplar

---

5. Funcionamiento del préstamo

Proceso general

```
USUARIO: "Quiero 1984"
↓
SISTEMA: Comprueba disponibilidad
    ├── Disponible:
    │   - Crea préstamo
    │   - Marca ejemplar como no disponible
    │
    └── No disponible:
        - Muestra mensaje de error
```

Código del préstamo

```java
public static Prestec addPrestec(Ejemplar ejemplar, Persona persona, ...) {
    Ejemplar ejemplarDB = session.find(Ejemplar.class, ejemplar.getId());

    if (ejemplarDB != null && ejemplarDB.isDisponible()) {
        Prestec prestec = new Prestec(ejemplarDB, persona, ...);
        ejemplarDB.setDisponible(false);
        session.persist(prestec);
        session.merge(ejemplarDB);
        return prestec;
    } else {
```

```
            System.out.println("❌ No se puede prestar: no disponible");
            return null;
        }
    }
```

Devolución del libro

```
public static void devolverPrestamo(Long prestamoId, LocalDate fecha) {
    Prestec prestec = session.find(Prestec.class, prestamoId);
    prestec.setActivo(false);
    prestec.setFechaDevolucionReal(fecha);
    prestec.getEjemplar().setDisponible(true);
}
```

---

6. Preguntas comunes y respuestas

¿Por qué necesitamos Hibernate?

Porque evita escribir SQL manualmente y reduce errores, permitiendo trabajar directamente con objetos Java.

¿Qué son `@Entity` y `@Id`?

Son anotaciones que indican a Hibernate que una clase es una tabla y que un campo es su clave primaria.

¿Por qué existen Libro y Ejemplar?

Para separar la **obra intelectual** de la **copia física**. Un libro puede tener muchas copias.

¿Qué pasa si un libro ya está prestado?

El sistema lo detecta y no permite crear un nuevo préstamo.

¿Cómo sabe el sistema qué libros hay en cada biblioteca?

Cada Ejemplar está asociado a una Biblioteca.

¿Para qué sirven las consultas HQL?

Para generar informes como libros disponibles, préstamos activos o autores por libro.

Ventajas frente a un sistema en papel

- Automático

- Rápido
- Seguro
- Sin pérdidas
- Genera informes fácilmente

---

7. Resumen para explicar en 1 minuto

> Hemos creado un sistema de gestión de biblioteca digital usando Hibernate. Permite registrar libros, autores y usuarios, gestionar préstamos y devoluciones, y generar informes. Hibernate se encarga de la base de datos sin necesidad de escribir SQL. Lo más importante es entender la diferencia entre Libro (la obra) y Ejemplar (la copia física).

---

8. Posibles preguntas del profesor

¿Qué fue lo más difícil?

Entender y modelar correctamente las relaciones, especialmente entre Libro y Ejemplar.

¿Qué has aprendido?

A usar Hibernate, diseñar bases de datos reales y aplicar lógica de negocio.

Si te pide explicar código

- Di qué hace el método
- Explica la lógica principal
- Menciona validaciones importantes

Si algo no lo sabes con exactitud:

> "Esta parte la implementé siguiendo ejemplos, aunque no conozco todos los detalles internos."

Recuerda: Lo importante es demostrar que entiendes el concepto general y que has trabajado en el proyecto.

¡Tú puedes! 🔆