Universidad de Oviedo

EPI GIJÓN

Universidad de Oviedo

Object Relational Mapping (ORM)

En Java definido como JPA

Enol García González
Universidad de Oviedo
10 de noviembre de 2025

Contenidos

ORM

- Object Relational Mapping (ORM) es una técnica de programación que consiste en mapear las estructuras de una base de datos relacional como objetos dentro de una programación OO.
- La principal ventaja de los ORMs es que facilitan mucho el manejo de las bases de datos y aceleran el desarrollo.
- Es un paradigma diferente para la gestión de la base de datos que abstrae la ejecución directa de las consultas. Lo que normalmente hacemos al trabajar con tecnologías con JDBC.

Desventajas

No todo es perfecto en los ORMs, también tienen algunas desventajas:

- Dependiendo de la implementación pueden acoplar en exceso la base de datos a la arquitectura de nuestra aplicación.
- La abstracción de la comunicación con la base de datos hace que como programadores tengamos menos control sobre las consultas que se envían a la base de datos.
- Son menos eficientes en términos de memoria y CPU.

Implementaciones

Todos los lenguajes orientados a objetos tienen diferentes librerías que implementan esta técnica de programación:

- Python tiene SQLAlchemy y peewee
- NodeJS tiene Sequelize
- C# tiene NHibernate y ADO.NET

- En java hay una especificación oficial que se conoce coo JPA. Esta especificación tiene diferentes implementaciones como EclipseLink o Hibernate.

Conceptos importantes de JPA

- En el modelo de la aplicación hay que crear una Clase de modelo por cada Tabla que tengamos en nuestra aplicación.
- Las clases pueden contener 3 tipos de atributos:
  - Los atributos **naturales** son aquellos que definen al objeto. Por ejemplo en un alumno, los atributos naturales serían su dni, nombre y apellido.
  - Los atributos **accidentales** son aquellos que surgen por cómo se representa la información en la base de datos. Por ejemplo, si el alumno esta relacionado con la tabla matrícula, habrá un atributo que almacene la lista de matriculas del alumno.
  - Los atributos **transitivos** son aquellos atributos que no se guardan en la base de datos, pero en Java nos interesa tenerlos almacenados en el objeto. Se suelen utilizar para atributos calculados, por ejemplo la nota media.

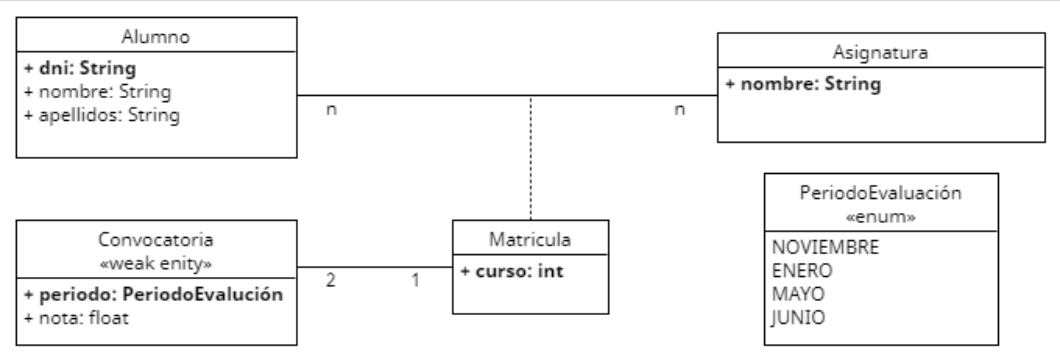
Conceptos importantes de JPA

- Los elementos de JPA se configuran de forma muy similar a cualquier framework moderno: con anotaciones sobre las clases y atributos o poniendo esa información en un fichero orm.xml.
- Es necesario incluir la anotación @Entity para representar que una clase representa una entidad de la base de datos. Si no no se guardará.
- Los atributos transitivos se deben marcar con una etiqueta @Transient.
- Los atributos accidentales de la relación se marcan con las etiquetas @ManyToMany, @ManyToOne, @OneToMany o @OneToOne según el tipo de relación
- Los atributos naturales no es necesario anotarlos, pero se pueden anotar con @Basic y @Column para personalizarlos. Por ejemplo, marcarlos como únicos o modificar el nombre con el que se guardarán en la base de datos.

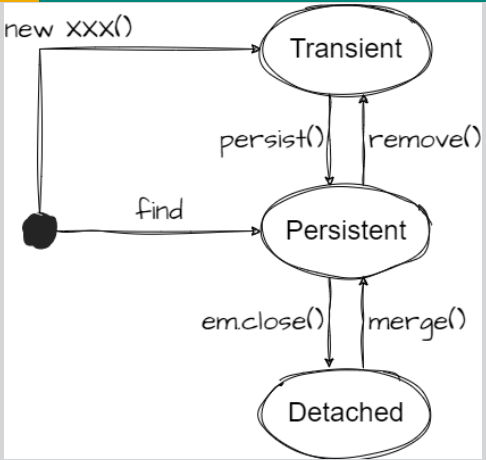
Conceptos importantes de JPA

- Con las clases están etiquetadas, se puede configurar el proyecto para que vacíe la base de datos y cree tablas nuevas a partir de las etiquetas o para que utilice una base de datos existente.
- Esta configuración se hace en un fichero con el nombre persistence.xml, donde también se incluye otra información como los datos de conexión.
- Todo esto funciona gracias a 2 interfaces importantes de JPA:
  - **EntityManager** es la responsable de toda la comunicación con la base de datos. Es la interfaz a la que le pediremos que nos ejecute consultas o que nos guarde objetos
  - **EntityManagerFactory** guarda la información del fichero persistence.xml para saber con que configuración crear las implementaciones de EntityManager.

Ejemplo básico



Estados en JPA



N-capas con JPA

- JPA nos facilita interactuar con la base de datos gracias a que mapea las tablas a objetos, pero eso no significa que nos olvidemos de arquitectura de proyecto.
- Con JPA también se recomienda la arquitectura n-capas:
  - La capa de persistencia se encargará de ejecutar las consultas y dar al servicio los objetos que solicite.
  - El único cambio en la persistencia es que no hace falta el método update.
  - En la capa de persistencia tendremos un DAO/Repositorio por cada una de las tablas.
  - La capa de servicio hace las operaciones de lógica de negocio.

N-capas con JPA

- Para que lo que cambiemos en la capa de negocio se actualice automáticamente en la base de datos es necesario que esta capa esté dentro del contexto de persistencia.
- El patrón TransactionScript separa cada método de las interfaces de servicio en una clase (script) por separado. Esta clase tiene un sólo método público.
- Se puede definir una plantilla para abrir y cerrar el contexto de persistencia con cada llamada al servicio.
- La presentación no sabe nada sobre que se usa JPA ni sobre nada detras de la interfaz de servicio.

Ejemplo

Ejemplo final para ver como se relaciona la arquitectura.