índice

Tema 6.

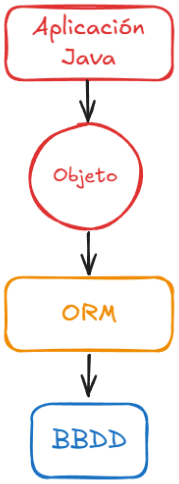
Definición del mapeo objeto relacional

Java lenguaje orientado a objetos, puede ser representado → gráfico de objetos.

Base de datos relacional se representa --> filas y columnas

ORM (Mapeo Objeto Relacional): Framework que facilita el almacenamiento de los objetos a una base de datos relacional.

Un objeto plano es el equivalente a una fila en una base de datos relacional.



Ventajas o inconvenientes del mapeo objeto relacional

Java tiene una API para la conexión a la BBDD: Los drivers de conexión JDBC (Java Database Connectivity).

La conexión a la BBDD en Java mediante JDBC necesita que los programadores conozcan bien la BBDD para hacer las consultas SQL y las relaciones.

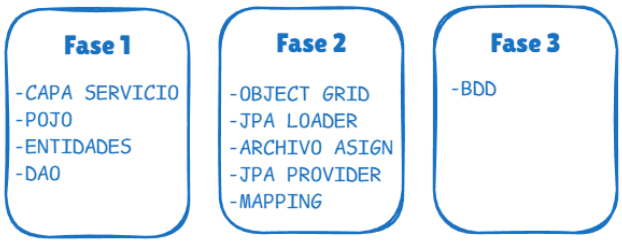
La ventaja de usar un enfoque orientado a objetos en Java, permite aplicar conceptos como abstracción, herencia, composición, identidad.. a través de frameworrks ORM (Mapeo Objeto Relacional). Facilitan el manejo de datos sin depender de SQL

Ventajas ORM	INCONVENIENTES
Eficiencia	**El mapeado automático** de BBDD consume muchos recursos
Desarrollo orientado a objetos POO	**La sintaxis de los ORM**
Manejabilidad	
Facilidad para introducir nuevas funciones	

Fases del mapeo objeto relacional

Arquitectura funcional de un framework ORM. Fases del acceso a datos de mapeo objeto-relacional (ORM) utilizando JPA (Java Persistence API).

- Fase 1: Datos del objeto.
  - POJO (Plain Old Java Objects) clases simples de Java.
  - Clases implementación, las clases e interfaces de la capa de negocio (capa de servicio)
  - Clases DAO (Data Access Object)
  - Contiene métodos como crearObjeto(), encontrarObjeto(), borrarObjeto()
- Fase 2: Persistencia o mapeo.
  - Proveedor JPA: Librería hace posible funcionalidad JPA: javax.persistence
  - Archivo de Asignación: fichero XML almacena la configuración de la asignación de los datos de una clase JAVA (POJO) y los datos reales de la BBDD relacional
  - JPA Cargador: funciona como una memoria caché, cargará los datos de la BBDD
  - Reja de Objetos: lugar donde se almacenan temporalmente una copia de los datos de nuestra BBDD relacional. Se le llama Objeto Grid , Todas las consultas pasan por aquí antes de ir a la base de datos principal.
- Fase 3: Fase de datos relacionales.
  - Después de pasar por la reja de objetos y todo haya ido bien, se irá directamente a la BBDD. Hasta entonces estará en caché



Herramientas ORM

Según el lenguaje de programación que utilicemos hay diferentes frameworks

- EBEAN:
  - **Soporte BBDD:** h2, Postgress, Mysql, PostGis, MariaDB, SQL Server, Oracle, SAP, etc
  - **Multiples niveles de abstracción:** Consultas de tipo ORM mezcladas con SQL, además de consultas DTO.
  - **Beneficios**: Evita automáticamente N+1, usa caché de tipo L2 para reducir carga BBDD, realiza consultas mezclando BBDD y cacheando, ajusta automáticamente las consultas ORM, contiene tecnología Elasticsearch para caché L3, etc.
- **IBATIS:** ORM que aparece de la mano de Apache software Foundation. Tiene soporte para Java y .Net.
  - Dividir la capa de persistencia en : capa de abstracción, capa de framework persistente y capa de Driver.
  - Facilidad de interactuar con objetos y los datos de las bases de datos relacionales
  - Abstracción a nivel de la capa de persistencia de objetos

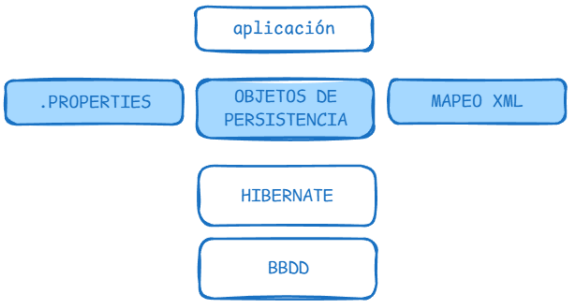
- **HIBERNATE:** es el ORM más extendido y más usado. Disponible para lenguaje Java y también para .Net (Hibernate). Facilita el mapeo relacional de los distintos objetos entre una base de datos relacional y el modelo de la aplicación; se apoya en un .xml establece las relaciones.
  - **Simplicidad**: al disponer de un solo .xml para establecer relaciones, es sencillo e intuitivo para consultar cualquier tipo de relación entre entidades o atributos.
  - **Robusto**: muchas características adaptadas al lenguaje Java: colecciones, herencia, abstracción, orientación a objetos, etc. En la capa de abstracción ofrece una propia capa de consultas SQL llamada **HQL,** facilitar la sintaxis y mejorar la eficiencia

Definición de la arquitectura de Hibernate

Un servicio de alta eficiencia mapeo objeto-relacional.

La arquitectura de Hibernate incluye distintos objetos como son los de persistencia, sesiones, transacciones entre otros

Hibernate usa el **principio de reflexión** Java, permite el análisis y la modificación de los atributos y características de las distintas clases en tiempo de ejecución.



La capa de aplicación e Hibernate están unidas por los Objetos de persistencia; porque en una parte específica de la aplicación se da cierta conversión (fichero mapeo), donde la información fluye y es mapeada desde ficheros persistentes a la base de datos.

En la capa Hibernate donde se realiza la conexión con el driver, también donde se cargan las distintas configuraciones Hibernate y las entidades.

Componentes de Hibernate

- SessionFactory Object:
  - Uso objetos tipo session
  - Se instancia a partir de un archivo de configuración
  - Se utiliza un SessionFactory por cada base de datos en la aplicación.
- Session Object
  - Establece una conexión directa con la base de datos relacional
  - Es un objeto ligero que interactúa con la base de datos.
  - No debe permanecer abierto mucho tiempo por razones de seguridad.
- Transaction Object
  - Maneja las transacciones con la base de datos de manera directa.
  - Es opcional; se pueden definir bloques transaccionales manualmente.
  - Permite persistir operaciones o realizar un rollback si es necesario.

- Query Object
  - Permite realizar consultas a la base de datos utilizando SQL o Hibernate Query Language (HQL).
  - Facilita la vinculación de parámetros, restricciones y ejecución de consultas de manera dinámica.
- Criteria Object
  - desaparece el uso de SQL nativo, permitiendo realizar consultas mediante objetos Java.
  - Utiliza funciones de Hibernate que se traducen a sentencias SQL.

Instalación de Hibernate

Métodos de Instalación:

- **Añadir el archivo .jar:** Se puede agregar manualmente el archivo .jar de la versión específica de Hibernate descargada.
- **Usar Maven:** En proyectos web con gestor de dependencias Maven, se puede agregar la dependencia de Hibernate en el archivo pom.xml. Al compilar, Maven descargará automáticamente la librería.

1. Instalación con Spring Boot:
   - Spring Boot, agregar Hibernate como dependencia.
   - Se utiliza la URL https://start.spring.io/
   - Al buscar "Hibernate", se puede añadir directamente Spring Data JPA y Hibernate.
2. Ejemplo de Dependencia:
   - La dependencia para agregar Hibernate con Spring Boot es:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

3. Alternativa sin Spring Boot:
   - Si se desea agregar Hibernate sin Spring Boot, se puede visitar la web oficial de Hibernate para encontrar la última versión de la dependencia adecuada.

Configuración de Hibernate

- **Application.properties** pondremos la comunicación con la base de datos
  - framework con la base de datos instalada en este caso la BBDD es H2

```
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

  - indicamos que habilitamos las trazas de tipo SQL, y que se muestren en formato SQL

```
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

  - habilitamos los logs de Hibernate a true; Podremos ver un log extenso sobre los errores que se vayan dando.

```
spring.jpa.properties.hibernate.generate_statistics=true
logging.level.org.hibernate.type=trace
logging.level.org.hibernate.stat=debug
```

  - Permitimos a Hibernate crear manualmente distintas entidades que queremmos cerar en un schema.sql

```
spring.jpa.hibernate.ddl-auto=none
```