Proyecto FPMADdigital

Recursos digitales y multimedia para Formación Profesional

Comunidad de Madrid

Dirección General de Educación Secundaria, Formación Profesional y Régimen Especial

CONSEJERÍA DE EDUCACIÓN, UNIVERSIDADES, CIENCIA Y PORTAVOCÍA

Unión Europea

Fondo Social Europeo

"El FSE invierte en tu futuro"

Financiado como parte de la respuesta de la Unión a la pandemia de COVID-19

FPMADdigital
Recursos digitales y multimedia para Formación Profesional

CFGS Desarrollo de Aplicaciones Multiplataforma

módulo profesional

0486 - Acceso a Datos

unidad didáctica

03 Procesamiento de BBDD Relacionales

resultados de aprendizaje

02 Desarrolla aplicaciones que gestionan información almacenada en bases de datos relacionales identificando y utilizando mecanismos de conexión.

03 Gestiona la persistencia de los datos identificando herramientas de mapeo objeto relacional (ORM) y desarrollando aplicaciones que las utilizan.

Comunidad de Madrid
Dirección General de Educación Secundaria, Formación Profesional y Régimen Especial
CONSEJERÍA DE EDUCACIÓN, UNIVERSIDADES, CIENCIA Y PORTAVOCÍA

Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
Financiado como parte de la respuesta de la Unión a la pandemia de COVID-19

Resultados de aprendizaje y unidades didácticas

Resultados de aprendizaje y unidades didácticas

RESULTADOS DE APRENDIZAJE						UNIDAD DIDÁCTICA
1	2	3	4	5	6	
X						1.- Ficheros, colecciones y data frames
				X		2.- Manejo de XML
	X	X				3.- Procesamiento de BBDD Relacionales
			X			4.- Uso de BBDD NoSQL
		X	X		X	5.- Programación de componentes de acceso a datos

Unidades didácticas y materiales asociados

Unidades didácticas y materiales multimedia

RRAA						UUDD	Material Multimedia
1	2	3	4	5	6		
X						1.- Ficheros, colecciones y data frames	1.1 Contenidos básicos 1.2 Ejemplos aplicados
				X		2.- Manejo de XML	2.1 Contenidos básicos 2.2 Ejemplos aplicados
	X	X				3.- Procesamiento de BBDD Relacionales	3.1 Contenidos básicos 3.2 Ejemplos aplicados
			X			4.- Uso de BBDD NoSQL	4.1 Contenidos básicos 4.2 Ejemplos aplicados
		X	X		X	5.- Programación de componentes de acceso a datos	5.1 Contenidos básicos 5.2 Ejemplos aplicados

Repositorios de materiales y prácticas

Repositorio de materiales y prácticas

Todos los proyectos mostrados, así como otros materiales utilizados en las unidades didácticas los podrás encontrar completos en:

https://github.com/joseluisgs/FP-NextGen-AccesoDatos

Cualquier error o propuestas de mejora se publicarán en el repositorio indicado.
Gracias por tu colaboración.

Contenidos

1. Bases de Datos Relacionales
2. Acceso a Base de Datos Relacionales
3. Desfase Objeto Relacional
4. ORMS
5. JPA

Bases de Datos Relacionales

Bases de Datos Relacionales

Desde que el ser humano existe, siempre hemos tenido la necesidad de almacenar la información, ya sea desarrollando la escritura, luego el papel, imprenta, libros, cintas perforadas, disco duro, etc.

A la misma vez que almacenamos la información debemos organizarla y procesarla ya sea en ficheros de acceso aleatorio, indexados o usando bases de datos.

Las **BBDD Relacionales** parte de un DBMS donde se cumple **el modelo relacional**:

- Una base de datos se compone de varias tablas, denominadas relaciones.
- No pueden existir dos tablas con el mismo nombre ni registro.
- Cada tabla es a su vez un conjunto de campos (columnas) y registros (filas). La clave primaria identifica un registro y las claves foráneas nos permiten expresar relaciones entre registros de distintas tablas.

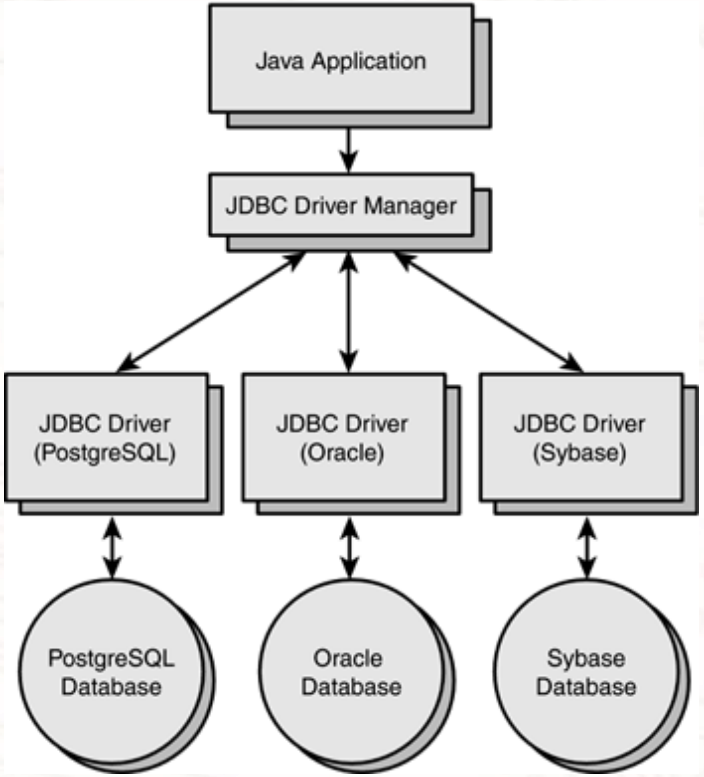
Acceso a Base de Datos Relacionales

Acceso a Base de Datos Relacionales

Para acceder a BBDD Relacionales o de otro tipo podemos hacer uso de **ODBC** o derivados. Open DataBase Connectivity (ODBC) es un estándar de acceso a las bases de datos desarrollado por SQL Access Group (SAG) en 1992.

En el contexto de JVM usaremos, **JDBC**, Java Database Connectivity, el cual es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación de JVM como son Java o Kotlin.

Con JDBC queda oculto la capa de acceso a base de datos para el desarrollador, por lo que éste sólo debe preocuparse de la aplicación y no de cómo se acceden a los datos gracias al **driver** propio de cada SGBD.



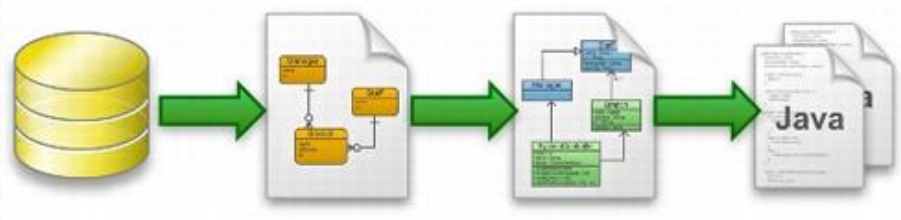
Desfase Objeto Relacional

Desfase Objeto Relacional

También conocido como **impedancia objeto-relacional**, se trata de la diferencia existente entre la programación orientada a objetos y las bases de datos relacionales. Nos referimos a:

- Lenguaje: se debe conocer dos lenguajes: el de programación y el de la base de datos.
- Paradigma de programación: tiene que haber una traducción del modelo entidad-relación de la base de datos al modelo orientado a objetos del lenguaje de programación.

Al trabajar con lenguajes orientados a objetos y bases de datos relacionales tenemos que estar continuamente descomponiendo los objetos para escribir la sentencia SQL para insertar, modificar o eliminar, o bien recomponer todos los atributos para formar el objeto cuando leamos algo de la base de datos.



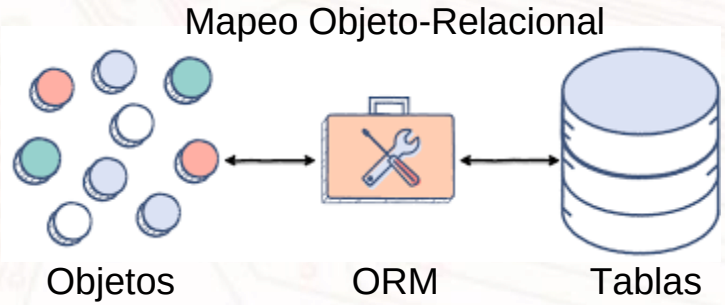
ORMS

ORMS

Object Relational Mapping (ORM) es la herramienta que nos sirve para transformar representaciones de datos de los Sistemas de Bases de Datos Relacionales, a representaciones (Modelos) de objetos.

Las herramientas ORM, actúan como un puente que conecta las ventajas de los RDBMS con la buena representación de estos en un lenguaje Orientado a Objetos.

Podemos partir de un modelo de objetos y obtener su representación relacional o viceversa, es el ORM el que **se encarga del mapeado entre los dos contextos** haciendo esta labor trasparente al programador/a.



ORMS

Algunos ORMS:

Hibernate: para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación.

Java Persistence API (JPA): es una especificación para la persistencia de objetos Java a cualquier base de datos relacional. Hibernate o SpringData implementa y deja utilizar JPA, ampliando la forma de utilizarlo.

Exposed: ORM de Jetbrains para Kotlin que permite realizar el mapeo en base a DSL o usando el patrón DAO.

ROOM: librería oficial de Google que actúa como ORM en el mundo del desarrollo Android para SQLite.

JPA

JPA

Los **mapeos de entidades** se hacen sobre clases usando anotaciones.

@Entity: indica que la clase es una tabla en la base de datos.

@Table: indica el nombre de la tabla.

@Id: indica que un atributo es la clave.

GenerationType.AUTO: mejor estrategia en función del dialecto SQL configurado.

GenerationType.SEQUENCE: espera usar una secuencia SQL para generar los valores de todas las claves.

GenerationType.IDENTITY: puede ser autonumérica (MariaDB) o siguiendo otra política que asegure que esa clave es única

GenerationType.TABLE: usa una tabla extra en nuestra base de datos. Tiene una tabla por entidad y una columna con el valor de la clave a usar.

JPA

Para el **mapeo de las relaciones**, además de crear el correspondiente objeto que permita mantener la relación entre las clases (de forma bidireccional), éstos atributos deben ser mapeados según convenga:

@OneToOne Indica que el objeto es parte de una relación ***1-1***

@ManyToOne Indica que el objeto es parte de una relación ***N-1***

@OneToMany Indica que el objeto es parte de una relación **1-N**

@ManyToMany Indica que el objeto es parte de una relación ***N-M***

En el otro lado de la relación indicaremos el tipo de relación acompañado de la anotación ***@MappedBy*** añadiendo el atributo de la otra clase donde se especifica toda la información sobre el mapeo para un acceso bidireccional.

JPA

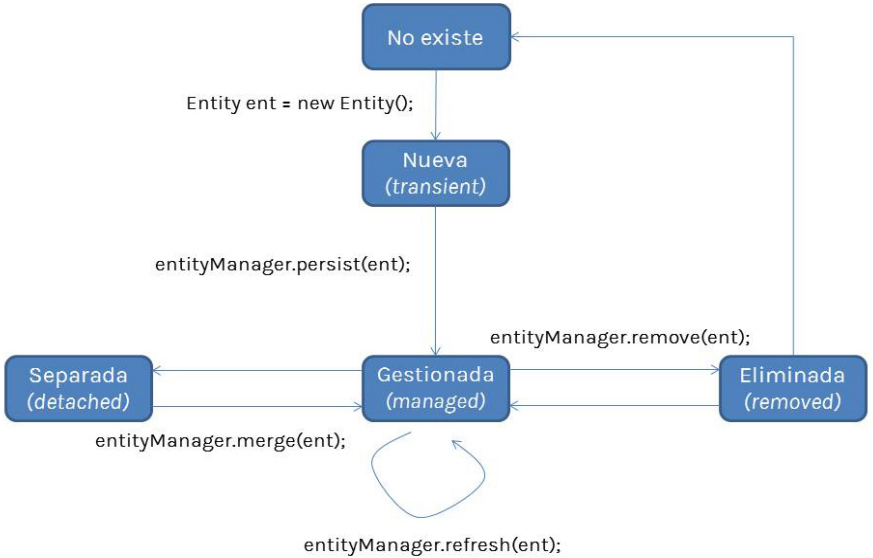
El ciclo de vida de una entidad en JPA en la misma sesión es controlado por el **EntityManager**.

transient (nueva): la entidad acaba de ser creada (posiblemente con el operador new) y aun no está asociada al contexto de persistencia. No tiene representación en la base de datos.

managed, persistent: la entidad tiene un identificador y está asociada al contexto de persistencia. Puede estar o no estar almacenada en la base de datos.

detached: la entidad tiene un identificador, pero no está asociada al contexto de persistencia (normalmente), porque hemos cerrado el contexto de persistencia.

removed: la entidad tiene un identificador y está asociada al contexto de persistencia, pero tiene programada su eliminación.



JPA

Para manejar **relaciones y transacciones** de datos debemos tener en cuenta que:

@orphanRemoval: marca a un hijo para ser eliminado, si no está referenciado por un padre.

Sobre las transiciones en cascada existen:

ALL: se propagan todos los cambios de estados.

PERSIST: se propagan las operaciones de persistencia.

MERGE: se propagan las operaciones merge.

REMOVE: se propagan las eliminaciones.

REFRESH: se propagan las actualizaciones.

DETACH: se propagan las separaciones.

JPA

Para la **herencia** tenemos:

@MappedSuperclass, la clase base no será trasladada a la base de datos, pero estará en memoria para su uso.

@Inheritance(strategy = InheritanceType.SINGLE_TABLE), para todas las entidades que participen en la jerarquía de herencia, solamente tiene que crear una tabla en la base de datos.

@Inheritance(strategy = InheritanceType.JOINED), habrá una tabla para la entidad base de la jerarquía. Tendrá todos los atributos de la clase base y una tabla para cada entidad extendida de la jerarquía. Tendrá una referencia a la entidad base, y los atributos propios.

@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS), cada entidad extendida tendrá los atributos de la clase base y los suyos propios (como en @MappedSuperclass), pero también existirá una tabla para la clase base.

JPA

Para hacer consultas podemos hacer uso de **JPQL** permitiéndonos hacer uso de parámetros nombrados o con índice.

JPA nos provee de dos interfaces, ***Query*** (no va tipado) y **TypedQuery** (el resultado va tipado), que se obtienen directamente desde el EntityManager.

Joins explícitos, JPQL permite realizar los mismos JOIN que en SQL (JOIN, INNER JOIN, LEFT | RIGHT JOIN)

Joins Impícitos. Un JOIN implícito se realiza siempre a través de alguna de las asociaciones de una entidad, navegando a través de sus propiedades de los objetos mapeados.

JPQL también nos permite ***lanzar consultas*** de actualización de datos, o de borrado, pero no insercción.

Conclusiones

¡Vamos con la práctica!

"La programación es una carrera entre los desarrolladores, intentando construir mayores y mejores programas a prueba de idiotas, y el universo, intentanto producir mayores y mejores idiotas. Por ahora va ganando el Universo"

- Rich Cook

Comunidad de Madrid
Dirección General de Educación Secundaria, Formación Profesional y Régimen Especial
CONSEJERÍA DE EDUCACIÓN, UNIVERSIDADES, CIENCIA Y PORTAVOCÍA

Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
Financiado como parte de la respuesta de la Unión a la pandemia de COVID-19