Asignatura

Acceso a datos

UNIVERSAE
Instituto Superior de FP

Asignatura

Acceso a datos

UNIDAD 4

Correspondencia objeto-relacional

UNIVERSAE
Instituto Superior de FP

Correspondencia objeto-relacional

¿Qué és?

- Proceso para realizar un mapa que describe la relación entre una clase en el lenguaje de programación y una tabla en la base de datos.
- Las siglas son ORM (Objet Relational Mapping)

Ventajas

- Facilita el desarrollo de la aplicación
- Desarrollos orientado a objetos
- Facilidad de gestión entre ambos sistemas
- Separa la aplicación del tipo de base de datos

Inconvenientes

- Exceso de consumo de recursos.
- Para un modelo de datos simple no tiene beneficio ninguna
- No hay mecanismos para optimizar

Hibernate

- Framework para Java para aplicar la correspondencia objeto-relacional
- El más usado.
- Gestiona la conexión y sesiones con las bases de datos
- Tiene sus propios lenguaje HQL y JPQL

Instalación

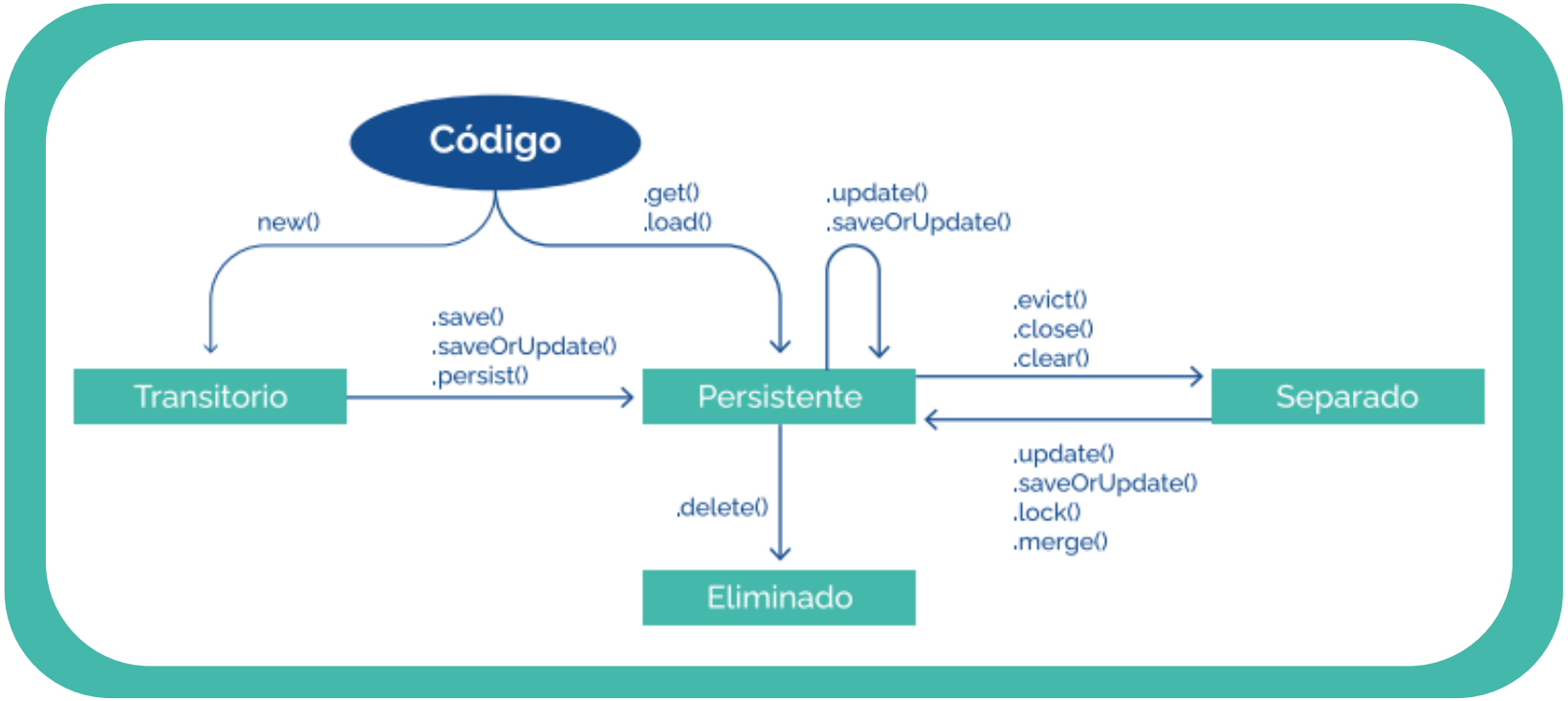
- La versión más reciente 6.1
- Para usarlo es necesario asociar las librerías al proyecto.
- Se puede añadir las dependencias con Maven
- (Opcional) usar un entorno de desarrollo adaptado a hibernate. NetBeans o añadir framework Jboss tools a Eclipse
- Añadir driver de la base de datos

Ficheros importantes

- hibernate.cfg.xml
- Ficheros ingeniería inversa .reveng.xml
- POJOs
- Ficheros de correspondencia .hbm.xml
- HibernateUtil.java

Sesiones y estados de los objetos

- La sesión contiene la conexión a la base de datos
- Gestiona las transacciones
- Los objetos, dependiendo si están cargados en sesión o no, pueden tener los estados: Transitorio, Persistente, Separado, Eliminado
- Los métodos aplicados en la sesión cambian los estado de los objetos



Ejemplo. Correspondencia (Mapeo)

```java
public class Clientes implements java.io.Serializable {

    private Integer id;
    private String nombre;
    private String email;
    private String telefono;
    private String direccion;
    private Set ordens = new HashSet(0);

    public Clientes() {
    }

    public Clientes(String nombre, String email,
                String telefono, String direccion) {
        this.nombre = nombre;
        this.email = email;
        this.telefono = telefono;
        this.direccion = direccion;
    }

    public Clientes(String nombre, String email,
                String telefono, String direccion, Set ordens) {
        this.nombre = nombre;
        this.email = email;
        this.telefono = telefono;
        this.direccion = direccion;
        this.ordens = ordens;
    }

    public Integer getId() {
        return this.id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    // .... Resto de get y set
}
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class catalog="ventas" name="com.ventas.pojos.Clientes" optimistic-lock="none" table="clientes">
        <id name="id" type="java.lang.Integer">
            <column name="id"/>
            <generator class="identity"/>
        </id>
        <property name="nombre" type="string">
            <column length="100" name="nombre" not-null="true"/>
        </property>
        <property name="email" type="string">
            <column length="100" name="email" not-null="true"/>
        </property>
        <property name="telefono" type="string">
            <column length="15" name="telefono" not-null="true"/>
        </property>
        <property name="direccion" type="string">
            <column length="65535" name="direccion" not-null="true"/>
        </property>
        <set fetch="select" inverse="true" lazy="true" name="ordens" table="orden">
            <key>
                <column name="cliente_id" not-null="true"/>
            </key>
            <one-to-many class="com.ventas.pojos.Orden"/>
        </set>
    </class>
</hibernate-mapping>
```

Ejemplo. Uso y configuración

```java
public static void main(String[] args) {

    Clientes cliente = new Clientes();
    cliente.setNombre("Maria");
    cliente.setTelefono("600888999");
    cliente.setEmail("maria@email.com");
    cliente.setDireccion("C/ Garcilaso, nº 12, 8ªA, Madrid");

    Productos producto = new Productos();
    producto.setNombre("Camiseta");
    producto.setDescripcion("Camiseta de color verde y talla unica");
    producto.setPrecio(10.5F);

    Session session = null;

    try {
        session = HibernateUtil.getSessionFactory().openSession();
        session.beginTransaction();

        session.save(cliente);
        session.save(producto);

        session.getTransaction().commit();
        session.close();

    } catch (Exception e) {
        e.printStackTrace(System.err);
        if(session != null)
            session.getTransaction().rollback();

    } finally {
        HibernateUtil.close();
    }

}
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.bytecode.use_reflection_optimizer">false</property>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.password">usuarioVentas</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3360/ventas</property>
        <property name="hibernate.connection.username">usuarioVentas</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.hbm2ddl.auto">none</property>
        <property name="hibernate.search.autoregister_listeners">true</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.validator.apply_to_ddl">false</property>

        <mapping resource="com/ventas/pojos/Orden.hbm.xml"/>
        <mapping resource="com/ventas/pojos/Clientes.hbm.xml"/>
        <mapping resource="com/ventas/pojos/OrdenProductos.hbm.xml"/>
        <mapping resource="com/ventas/pojos/Productos.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Operatividad

A partir de la sesión

- Métodos get(), load(), save(), delete(), ..

Lenguaje HQL y JPQL

- Lenguajes basados en SQL
- Uso de la interfaz Query.
- Parecido a JDBC

Mediante SQL

- Uso de sentencias de SQL
- Uso del método .createNativeQuery(SQL)
- Pertenece a la sesión

```
Session session = null;

try {
    session = HibernateUtil.getSessionFactory().openSession();

    Query query = session.createQuery("FROM Clientes WHERE id = :id");

    query.setParameter("id", 1);

    List<Clientes> clientes = (List<Clientes>) query.getResultList();

    for (Clientes cliente : clientes) {
        System.out.print(cliente.getId() +" "+ cliente.getNombre()
                + " "+ cliente.getDireccion() + " "+cliente.getEmail()+"\n");
    }

    session.close();

} catch (Exception e) {
    e.printStackTrace(System.err);
} finally {
    HibernateUtil.close();
}
```

```
Session session = null;

try {
    session = HibernateUtil.getSessionFactory().openSession();

    List<Object[]> query =
            session.createNativeQuery("SELECT * FROM Clientes").list();

    for (Object[] cliente : query) {
        System.out.print(cliente[0]+" ");
        System.out.print(cliente[1]+" ");
        System.out.print(cliente[2]+" ");
        System.out.println(cliente[3]);
    }

    session.close();

} catch (Exception e) {
    e.printStackTrace(System.err);
} finally {
    HibernateUtil.close();
}
```

Correspondencia de la herencia

Herencia en ORM

- Problema. Las bases de datos relacionales no existe el modelo de datos para representar la herencia.
- El ORM si dispone de herramientas para mapear.
- Posibles soluciones:

Eliminación de subtipos

- Una tabla para la jerarquía
- Todos los campos de la herencia
- Es necesario un campo de clasificación (Tipo)
- Uso de <subclass> <discriminator> y discriminator-value

Eliminación de la jerarquía

- Una tabla para cada subclase
- Cada tabla tiene como PK la clave de la superclase
- Uso de <joined-subclass>

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class catalog="ventas" name="com.herencia.unatabla.Vehiculo"
                        table="vehiculo" discriminator-value="vehiculo">
        <id name="id" type="java.lang.Integer">
            <column name="id"/>
            <generator class="identity"/>
        </id>
        <discriminator column="tipo" type="string"></discriminator>
        <property name="marca" type="string">
            <column length="100" name="marca" not-null="true"/>
        </property>
        <property name="modelo" type="string">
            <column length="100" name="modelo" not-null="true"/>
        </property>
        <subclass name="com.herencia.unatabla.Coche" discriminator-value="coche">
            <property name="idCoche" type="int">
                <column length="10" name="id_coche"/>
            </property>
            <property name="numPuertas" type="int">
                <column length="2" name="num_puertas"/>
            </property>
        </subclass>
        <subclass name="com.herencia.unatabla.Moto" discriminator-value="moto">
            <property name="idMoto" type="int">
                <column length="10" name="id_moto"/>
            </property>
            <property name="numPlazas" type="int">
                <column length="2" name="num_plazas"/>
            </property>
        </subclass>
    </class>
</hibernate-mapping>
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class catalog="ventas" name="com.herencia.multipletabla.Vehiculo" table="vehiculo" >
        <id name="idVehiculo" type="java.lang.Integer">
            <column name="id_vehiculo"/>
            <generator class="identity"/>
        </id>
        <property name="marca" type="string">
            <column length="100" name="marca" not-null="true"/>
        </property>
        <property name="modelo" type="string">
            <column length="100" name="modelo" not-null="true"/>
        </property>
        <joined-subclass name="com.herencia.multipletabla.Coche">
            <key column="id_vehiculo"/>
            <property name="idCoche" type="int">
                <column length="10" name="id_coche"/>
            </property>
            <property name="numPuertas" type="int">
                <column length="2" name="num_puertas"/>
            </property>
        </joined-subclass>
        <joined-subclass name="com.herencia.multipletabla.Moto">
            <key column="id_vehiculo"/>
            <property name="idMoto" type="int">
                <column length="10" name="id_moto"/>
            </property>
            <property name="numPlazas" type="int">
                <column length="2" name="num_plazas"/>
            </property>
        </joined-subclass>
    </class>
</hibernate-mapping>
```

Resumen

1. Correspondencia objeto-relacional
2. Hibernate
3. Sesiones y estado de los objetos
4. Ejemplo. Correspondencia (Mapeo)
5. Ejemplo. Uso y configuración
6. Operatividad
7. Correspondencia de la herencia

UNIVERSAE

CHANGE YOUR WAY