MEDAC
Instituto Oficial de Formación Profesional

Anexo Práctica
Tema 7 y 8

Mapeo objeto-relacional y exploración

Hibernate + Consultas HQL

```
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
      <scope>runtime</scope>
</dependency>
```

Dependencias pom.xml	Fichero

```
<hibernate-configuration>
    <session-factory>
        <!-- Propiedades de conexión a la BBDD-->
        <property name="connection.driver_class">*********</property>
        <property name="connection.url">**************</property>
        <property name="connection.username">****</property>
        <property name="connection.password">****</property>

        <!-- Mapeo de entidades -->
       <mapping resource="cliente.hbm.xml"/>
   </session-factory>
</hibernate-configuration>
```

hibernate.cfg.xml	Fichero

```
package com.example.demo;

public class Cliente {

    private int IdCliente;
    private String nombre;
    private String apellido;
    private String email;
    private String direccion;
    private String telefono;


    public Cliente() {
    }

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

    public int getIdCliente() {
        return IdCliente;
    }

    public void setIdCliente(int
idCliente) {
        IdCliente = idCliente;
    }

    public String getNombre() {
        return nombre;
    }
```

```
public void setNombre(String nombre) {
        this.nombre = nombre;
    }

public String getApellido() {
        return apellido;
    }

    public void setApellido(String
apellido) {
        this.apellido = apellido;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email)
{
        this.email = email;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String
direccion) {
        this.direccion = direccion;
    }

    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String
telefono) {
        this.telefono = telefono;
    }
}
```

Clase persistente Cliente .java	Fichero : Cliente.java

```
<!-- Archivo de asignación: es un fichero XML donde se almacena la
configuración de la asignación de los
#datos de una clase JAVA (POJO) y los datos reales de la base de datos
relacional.-->


<hibernate-mapping>
    <class name="com.example.demo.Cliente" table="Clientes">
        <meta attribute="class-description">
            Esta clase contiene información de los clientes.
        </meta>
        <id name="idCliente" type="int" column="ID_Cliente">
            <generator class="native"/>
        </id>
        <property name="nombre" column="Nombre" type="string"/>
        <property name="apellido" column="Apellido" type="string"/>
        <property name="email" column="Email" type="string"/>
        <property name="direccion" column="Direccion" type="string"/>
        <property name="telefono" column="Telefono" type="string"/>
    </class>
</hibernate-mapping>
```

Carpeta : main/resources	Fichero : cliente.hbm.xml

```
/*Esta línea crea un SessionFactory de Hibernate.
new Configuration() crea un objeto de configuración de Hibernate.
.configure() carga la configuración de Hibernate desde el archivo
hibernate.cfg.xml por defecto.
.buildSessionFactory() construye el SessionFactory basado en esa
configuración. */

SessionFactory sessionFactory = new
Configuration().configure().buildSessionFactory();

/*Aquí se abre una sesión de Hibernate a través del SessionFactory. Una sesión
de Hibernate permite realizar operaciones con la base de datos.*/

Session session = sessionFactory.openSession();
Scanner scanner = new Scanner(System.in);

System.out.println("Inserte el código de cliente:");
Integer idCliente = scanner.nextInt();
System.out.print("Teléfono: ");
String telefono = scanner.next();
Query<Cliente> query = session.createQuery("update Cliente set telefono =
:nuevoTelefono where idCliente = :idCliente", Cliente.class);
query.setParameter("nuevoTelefono", telefono);
query.setParameter("idCliente", idCliente);

int status = query.executeUpdate();

if (status > 0) {
        System.out.println("Cliente actualizado con éxito.");
        session.getTransaction().commit();
        // Confirmar los cambios
} else {
        System.out.println("No se pudo actualizar el cliente.");
        session.getTransaction().rollback();
        // Revertir la transacción si no hay cambios
}
session.close();
sessionFactory.close();
```

Carpeta Ejemplo Actualizar cliente mediante Hibernate + HQL	Main.java

HQL

Ejemplos consultas HQL

```
Query query=session.createQuery("from Emp");
//here persistent class name is Emp
List list=query.list();
```

```
Query query=session.createQuery("from Emp");
query.setFirstResult(5);
query.setMaxResult(10);
List list=query.list();
//will return the records from 5 to 10th number
```

```
Query q=session.createQuery("update User set name=:n where id=:i");
q.setParameter("n","Udit Kumar");
q.setParameter("i",111);
```

```
Query query=session.createQuery("delete from Emp where id=100");
//specifying class name (Emp) not tablename
```

```
Query q=session.createQuery("select sum(salary) from Emp");
List<Integer> list=q.list();
System.out.println(list.get(0));
```