Guía de Laboratorio: Integración de Hibernate en Proyectos Java

Objetivo

Aprender a integrar Hibernate en una aplicación Java, mapear clases a tablas de base de datos y realizar operaciones CRUD.

Caso: Gestión de Clientes en una Tienda

1. Configuración de Hibernate

1.1 Dependencias Maven

Si usas Maven, agrega las siguientes dependencias en `pom.xml`:

```
<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>6.0.0.Final</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.33</version>
    </dependency>
</dependencies>
```

1.2 Archivo de configuración `hibernate.cfg.xml`

```
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/tienda</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password"></property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="show_sql">true</property>
    </session-factory>
</hibernate-configuration>
```

2. Creación de Clases

2.1 Clase `Cliente` (Entidad)

```
import jakarta.persistence.*;

@Entity
@Table(name = "clientes")
public class Cliente {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String nombre;
    private String email;
    private String telefono;

    public Cliente() {}

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

    // Getters y Setters
}
```

2.2 Clase `HibernateUtil` (Manejo de Sesión)

```
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        return new Configuration().configure().buildSessionFactory();
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

3. Operaciones CRUD

3.1 Clase `ClienteDAO`

```
import org.hibernate.Session;
import org.hibernate.Transaction;
import java.util.List;

public class ClienteDAO {
    public void agregarCliente(Cliente cliente) {
        Transaction tx = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();
            session.save(cliente);
            tx.commit();
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            e.printStackTrace();
        }
    }

    public List<Cliente> listarClientes() {
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            return session.createQuery("from Cliente", Cliente.class).list();
        }
    }

    public void actualizarCliente(int id, String nuevoEmail) {
        Transaction tx = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();
            Cliente cliente = session.get(Cliente.class, id);
            if (cliente != null) {
                cliente.setEmail(nuevoEmail);
                session.update(cliente);
                tx.commit();
            }
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            e.printStackTrace();
        }
    }

    public void eliminarCliente(int id) {
        Transaction tx = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();
            Cliente cliente = session.get(Cliente.class, id);
            if (cliente != null) {
                session.delete(cliente);
                tx.commit();
            }
        } catch (Exception e) {
            if (tx != null) tx.rollback();
```

```
            e.printStackTrace();
        }
    }
}
```

---

4. Prueba del CRUD

Clase `Main`

```
import java.util.List;

public class Main {
    public static void main(String[] args) {
        ClienteDAO clienteDAO = new ClienteDAO();

        // Agregar Cliente
        Cliente cliente1 = new Cliente("Juan Pérez", "juan@email.com",
"987654321");
        clienteDAO.agregarCliente(cliente1);

        // Listar Clientes
        List<Cliente> clientes = clienteDAO.listarClientes();
        clientes.forEach(cliente -> System.out.println(cliente.getNombre()));

        // Actualizar Cliente
        if (!clientes.isEmpty()) {
            clienteDAO.actualizarCliente(clientes.get(0).getId(),
"nuevo@email.com");
        }

        // Eliminar Cliente
        if (!clientes.isEmpty()) {
            clienteDAO.eliminarCliente(clientes.get(0).getId());
        }
    }
}
```

Caso 2: Gestión de Zoológico

2. Creación de Clases

2.1 Clase `Animal` (Entidad)

```
import jakarta.persistence.*;

@Entity
```

```
@Table(name = "animales")
public class Animal {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String nombre;
    private String especie;
    private int edad;

    public Animal() {}

    public Animal(String nombre, String especie, int edad) {
        this.nombre = nombre;
        this.especie = especie;
        this.edad = edad;
    }

    // Getters y Setters
}
```

2.2 Clase `HibernateUtil` (Manejo de Sesión)

```
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        return new Configuration().configure().buildSessionFactory();
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

---

3. Operaciones CRUD

3.1 Clase `AnimalDAO`

```
import org.hibernate.Session;
import org.hibernate.Transaction;
import java.util.List;

public class AnimalDAO {
    public void agregarAnimal(Animal animal) {
```

```
        Transaction tx = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();
            session.save(animal);
            tx.commit();
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            e.printStackTrace();
        }
    }

    public List<Animal> listarAnimales() {
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            return session.createQuery("from Animal", Animal.class).list();
        }
    }

    public void actualizarAnimal(int id, int nuevaEdad) {
        Transaction tx = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();
            Animal animal = session.get(Animal.class, id);
            if (animal != null) {
                animal.setEdad(nuevaEdad);
                session.update(animal);
                tx.commit();
            }
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            e.printStackTrace();
        }
    }

    public void eliminarAnimal(int id) {
        Transaction tx = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();
            Animal animal = session.get(Animal.class, id);
            if (animal != null) {
                session.delete(animal);
                tx.commit();
            }
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            e.printStackTrace();
        }
    }
}
```

---

4. Prueba del CRUD

Clase `Main`

```java
import java.util.List;

public class Main {
    public static void main(String[] args) {
        AnimalDAO animalDAO = new AnimalDAO();

        // Agregar Animal
        Animal animal1 = new Animal("Simba", "León", 5);
        animalDAO.agregarAnimal(animal1);

        // Listar Animales
        List<Animal> animales = animalDAO.listarAnimales();
        animales.forEach(animal -> System.out.println(animal.getNombre()));

        // Actualizar Animal
        if (!animales.isEmpty()) {
            animalDAO.actualizarAnimal(animales.get(0).getId(), 6);
        }

        // Eliminar Animal
        if (!animales.isEmpty()) {
            animalDAO.eliminarAnimal(animales.get(0).getId());
        }
    }
}
```