Licenciatura en Sistemas - Orientación a Objetos II – 2018

Prof. Titular: Lic. María Alejandra Vranić alejandravranic@gmail.com
facebook: alejandravranic

Prof. Ayudantes: APU Leandro Ríos leandro.rios.unla@gmail.com
facebook: Leandro Ríos

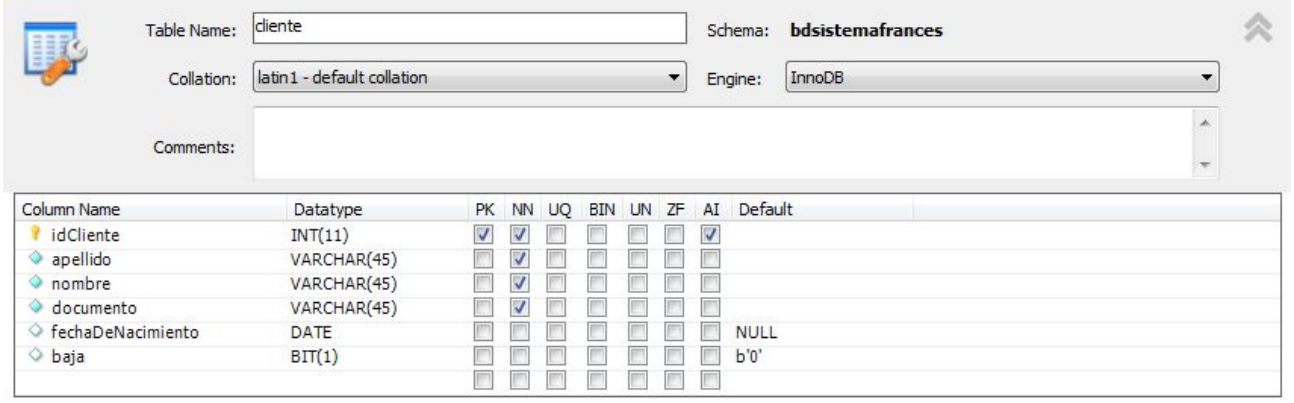
APU Gustavo Siciliano gussiciliano@gmail.com
facebook: Gus Siciliano

IDE: Eclipse
Persistencia de datos: MySQL
Bibliografía: ver programa Hibernate

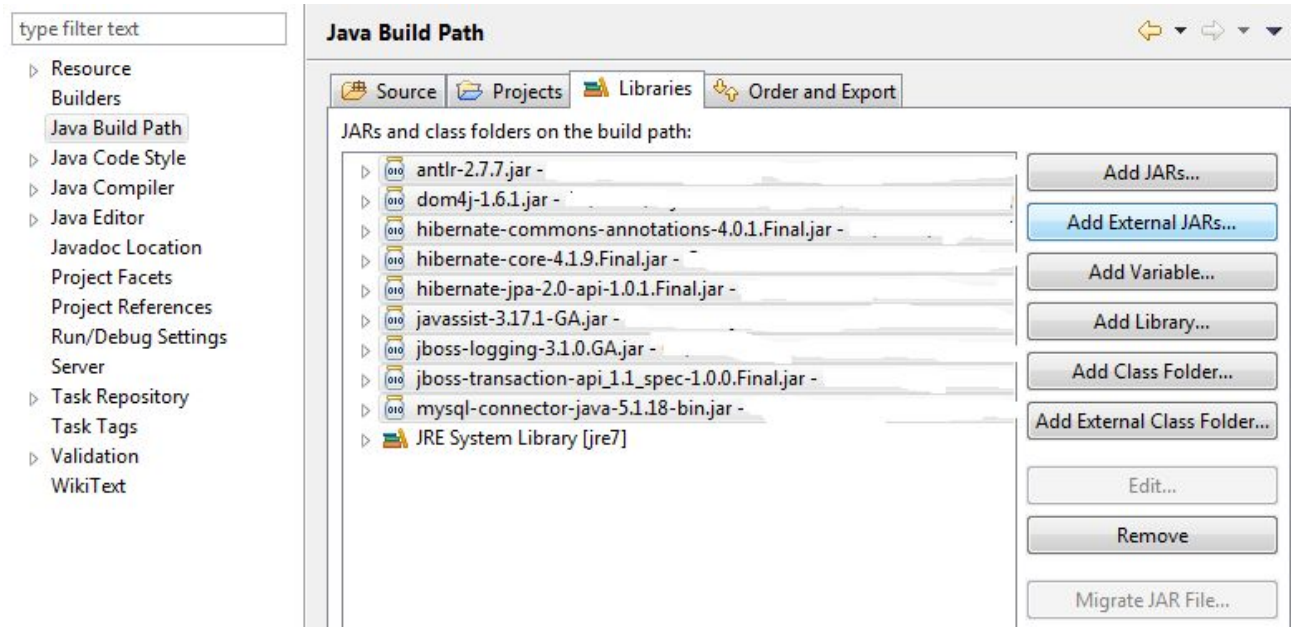
Framework Hibernate

Gavin King in 2001, crea Hibernate un framework Object-Relational Mapping (ORM) cuyo objetivo es la persistencia de objetos y consultas con un modelo de de base de datos relacional y de una Java Application.

En MySQL importar bd-hibernate-una-entidad.sql (la clave primaria debe ser autoincrementable)



Crear Java Project (\\objetos2\Hibernate-UnaEntidad)
guardar fuera de la carpeta proyecto, la carpeta lib ejemplo \\objetos2\lib , esta carpeta va contener todos los archivos .jar; cada vez que hacemos un proyecto vamos a tener que mapear las librerias.
Proyect → Properties
Java Build Path
Add External JARs
mapear la carpeta lib y hacer clic en el botón aceptar



nota:
Para bajar la librerías de hibernate hibernate-release-4.1.9.Final
http://sourceforge.net/projects/hibernate/files/hibernate4/4.1.9.Final/

y para el Connector/ODBC http://dev.mysql.com/downloads/connector/j/5.0.html

Crear los paquetes dao, datos, negocio, mapeos y test
En en el paquete datos crear la clase Cliente

```
package datos;

import java.util.GregorianCalendar;
import funciones.Funciones;

public class Cliente {
	private long idCliente;
	private String apellido;
	private String nombre;
	private int dni;
	private GregorianCalendar fechaDeNacimiento;
	private boolean baja;

	public Cliente(){} //siempre hay que implementar el contructor vacio

	public Cliente(String apellido, String nombre, int dni,
			GregorianCalendar fechaDeNacimiento) {//nunca va el id en el
constructor por ser autoincrementable
		super();
		this.apellido = apellido;
		this.nombre = nombre;
		this.dni= dni;
		this.fechaDeNacimiento = fechaDeNacimiento;
		this.baja=false;
	}

	public long getIdCliente() {
		return idCliente;
```

```
	}

	protected void setIdCliente(long idCliente) {//siempre va protected, para que no
sea modificado
		this.idCliente = idCliente;
	}

	public String getApellido() {
		return apellido;
	}

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

	public String getNombre() {
		return nombre;
	}

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


	public int getDni() {
		return dni;
	}


	public void setDni(int dni) {
		this.dni = dni;
	}


	public GregorianCalendar getFechaDeNacimiento() {
		return fechaDeNacimiento;
	}

	public void setFechaDeNacimiento(GregorianCalendar fechaDeNacimiento) {
		this.fechaDeNacimiento = fechaDeNacimiento;
	}

	public boolean isBaja() {
		return baja;
	}

	public void setBaja(boolean baja) {
		this.baja = baja;
	}

	public String toString(){
		return (idCliente+" "+apellido+" "+nombre+" DNI: "+dni+" F.de Nacimiento:
"+Funciones.traerFechaCorta(fechaDeNacimiento)+" "+baja);
	}

}
```

Crear dentro del paquete mapeos: src\mapeos\Cliente.hbm.xml

```
<<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

<class name="datos.Cliente" table="cliente">
    <id column="idCliente" name="idCliente">
      <generator class="identity"/>
    </id>
    <property column="apellido" name="apellido" type="string"/>
    <property column="nombre" name="nombre" type="string"/>
    <property column="dni" name="dni" type="int"/>
      <property column="fechaDeNacimiento" name="fechaDeNacimiento" type="calendar"/>
    <property column="baja" name="baja" type="boolean"/>
  </class>
</hibernate-mapping>
```

Crear el archivo xml dentro de src: Hibernate-UnaEntidad\src\hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="connection.url">jdbc:mysql://localhost/bd-hibernate-una-entidad</property>
    <property name="connection.username">root</property>
    <property name="connection.password">root</property>
    <property name="connection.pool_size">1</property>
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="show_sql">true</property>    <!-- en true muestra hql en consola-->
    <!--Mapeo Entidades   -->
      <mapping resource="mapeos/Cliente.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Dentro del paquete dao crear la clase `HibernateUtil`

```
package dao;
import org.hibernate.HibernateException;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;
import org.hibernate.service.ServiceRegistryBuilder;

public class HibernateUtil{
    private static SessionFactory sessionFactory;
    public static SessionFactory getSessionFactory() {
      try{
       if (sessionFactory == null) {
           Configuration configuration = new Configuration().configure();
           ServiceRegistryBuilder registry = new ServiceRegistryBuilder();
           registry.applySettings(configuration.getProperties());
           ServiceRegistry serviceRegistry = registry.buildServiceRegistry();
           sessionFactory = configuration.buildSessionFactory(serviceRegistry);
```

```
        }

       } catch (HibernateException he) {           System.err.println("ERROR en la
inicialización de la SessionFactory: " + he);
           throw new ExceptionInInitializerError(he);
       }
        return sessionFactory
}
}
```

Dentro del paquete dao crear la clase ClienteDao

```
package dao;

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

import datos.Cliente;


public class ClienteDao {
      private static Session session;
    private Transaction tx;

    private void iniciaOperacion() throws HibernateException {
        session = HibernateUtil.getSessionFactory().openSession();
        tx = session.beginTransaction();
    }

    private void manejaExcepcion(HibernateException he) throws HibernateException {
        tx.rollback();
        throw new HibernateException("ERROR en la capa de acceso a datos", he);
    }

    public int agregar(Cliente objeto) {
        int id = 0;
        try {
            iniciaOperacion();
            id = Integer.parseInt(session.save(objeto).toString());
            tx.commit();
        } catch (HibernateException he) {
            manejaExcepcion(he);
            throw he;
        } finally {
            session.close();

        }
        return id;
    }

    public void actualizar(Cliente objeto) throws HibernateException {
        try {
            iniciaOperacion();
            session.update(objeto);
            tx.commit();
        } catch (HibernateException he) {
```

```
            manejaExcepcion(he);
            throw he;
        } finally {
            session.close();
        }
    }

    public void eliminar(Cliente objeto) throws HibernateException {
        try {
            iniciaOperacion();
            session.delete(objeto);
            tx.commit();
        } catch (HibernateException he) {
            manejaExcepcion(he);
            throw he;
        } finally {
            session.close();
        }
    }

       public Cliente traerCliente(long idCliente) throws HibernateException {
        Cliente objeto = null;
        try {
            iniciaOperacion();
            objeto = (Cliente) session.get(Cliente.class, idCliente);
        } finally {
            session.close();
        }
        return objeto;
    }

    public Cliente traerCliente(int dni) throws HibernateException {
        Cliente objeto = null;

        try {
            iniciaOperacion();
            objeto = (Cliente) session.createQuery("from Cliente c where
c.dni="+dni).uniqueResult();
        } finally {
            session.close();
        }
        return objeto;
    }

    @SuppressWarnings("unchecked")
      public List<Cliente> traerCliente() throws HibernateException {
      List<Cliente> lista=null;
        try {
            iniciaOperacion();
            lista=session.createQuery("from Cliente c order by c.apellido asc c.nombre
asc").list();

        } finally {
            session.close();
        }

        return lista;
    }
```

```
}
```

Dentro del paquete negocio crear la clase ClienteABM

```
package negocio;

import java.util.GregorianCalendar;
import java.util.List;

import dao.ClienteDao;
import datos.Cliente;

public class ClienteABM {
    ClienteDao dao=new ClienteDao();
    public Cliente traerCliente(long idCliente){
          Cliente c= dao.traerCliente(idCliente);
          // implementar si c es null lanzar Exception
          return c;
       }
    public Cliente traerCliente(int dni){
          Cliente c= dao.traerCliente(dni);
          // implementar si c es null lanzar Exception
          return c;
       }
       public int agregar(String apellido, String nombre, int dni,
                   GregorianCalendar fechaDeNacimiento){
                //consultar si existe un cliente con el mismo dni, si existe arrojar la
Excepcion
                 Cliente c=new Cliente(apellido, nombre, dni,fechaDeNacimiento);

                return dao.agregar(c);
       }
       public void modificar(Cliente c){
                 /* implementar antes de actualizar que no exista un cliente
                 con el mismo documento a modificar
                  y con el mismo id, lanzar la Exception */
                dao.actualizar(c);
       }
```

public void eliminar(long idCliente){/*en este caso es física en gral. no se se aplicaría este caso de uso, si se hiciera habría que validar que el cliente no tenga dependencias*/

```
             Cliente c=dao.traerCliente(idCliente);
             //Implementar que si es null que arroje la excepción la Excepción
             dao.eliminar(c);
       }

       public List<Cliente> traerCliente(){return dao.traerCliente();}
}
```

En el paquete test

```
package test;

import java.util.GregorianCalendar;
import negocio.ClienteABM;

public class TestAgregarCliente {
	 public static void main(String[] args) {


	String apellido="tu apellido";
	String nombre="tu nombre";
	int documento=35000000;
	GregorianCalendar fechaDeNacimiento=new GregorianCalendar();//tu fecha de
nacimiento
	ClienteABM abm=new ClienteABM();
	long ultimoIdCliente= abm.agregar(apellido, nombre, documento,
fechaDeNacimiento);

	 }
}

package test;


import datos.Cliente;
import negocio.ClienteABM;

public class TestActualizarCliente {
	 public static void main(String[] args) {

	ClienteABM abm = new ClienteABM();
			long id=1;
	 //traer el obj a modificar
			Cliente c=abm.traerCliente(id);
			   System.out.println("Cliente a Modificar -->"+c);
	 //modificar por set los atributos
		c.setDni(35000001);
		abm.modificar(c); //update del objeto
		 int dni=35000001;
		Cliente cModif=abm.traerCliente(dni);
		System.out.println("Cliente Modificado -->"+cModif);

	 }
}
```

Queda pendiente implementar: Eliminar cliente y Traer todos los clientes