PRESA. UNIDAD 3. ACCESO MEDIANTE MAPEO RELACIONAL DE OBJETOS (ORM). HIBERNAR PARTE 1

PRESA. Acceso a Datos (ADA) (a distancia en inglés)

Unidad 3. ACCESO UTILIZANDO MAPEO RELACIONAL DE OBJETOS (ORM)

Parte 1. Acceso usando Hibernate clásico (ejemplo)

Abelardo Martinez

Basado y modificado de Sergio Badal (www.sergiobadal.com)

Año 2023-2024

Aspectos a tener en cuenta

Importante

Si buscas las soluciones navegando por Internet o preguntando al oráculo de ChatGPT te estarás engañando. Tenga en cuenta que ChatGPT no es infalible ni todopoderoso.

Es una gran herramienta para acelerar tu trabajo una vez que dominas una materia, pero utilizarla como atajo a la hora de adquirir habilidades y conocimientos básicos socava gravemente tu aprendizaje. Si lo utiliza para obtener soluciones o consejos por su cuenta, consulte también detenidamente las soluciones propuestas. Intenta resolver las actividades utilizando los recursos que hemos visto y la documentación ampliada que encontrarás en el "Aula Virtual".

Consejos para programar

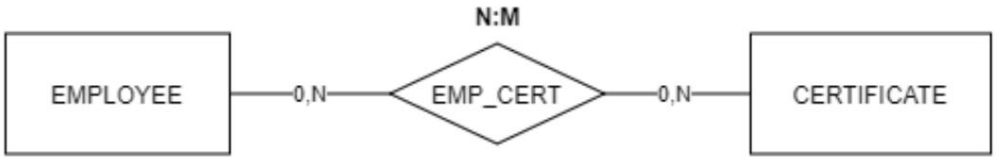
Recomendamos seguir los siguientes estándares de codificación:

- Una instrucción por línea.
- Agregue comentarios para que su código sea más claro y legible.
- Utilizar la notación húngara para reconocer el tipo de variables a primera vista.
- Recuerda que hay varias formas de implementar una solución, así que elige la que más te guste.
  Recomendamos encarecidamente utilizar soluciones basadas en buffer.

1. Modo consola. Administrar una base de datos relacional MySQL con Hibernate classic

Actividad (no evaluable)

Cree un proyecto Java para administrar este diagrama ER en MySQL utilizando los recursos de mapeo Maven e Hibernate CLASSIC.



ATENCIÓN: Utilice las excepciones adecuadas al acceder a bases de datos a través de Hibernate.

DEBES trabajar con Java, Maven e Hibernate, usando el estilo antiguo como se explicó esta semana (recursos de mapeo clásicos en lugar de usar anotaciones).

Crear base de datos en MySQL

Solución

```
CREAR BASE DE DATOS DBCertificados CONJUNTO DE CARACTERES utf8 COLLATE utf8_spanish_ci;

CREAR USUARIO mavenuser@localhost IDENTIFICADO POR 'ada0486';
CONCEDA TODOS LOS PRIVILEGIOS EN DBCertificates.* a mavenuser@localhost;

USE Certificados DB;

CREAR TABLA Empleado (
    empID           INTEGER NO NULO AUTO_INCREMENT,
    nombre de pila  VARCHAR(20),
    apellido        VARCHAR(20),
    salario         DOBLE,
    RESTRICCIÓN emp_id_pk CLAVE PRIMARIA (empID)
);

CREAR TABLA Certificado (
    ID de certificado       INTEGER NO NULO AUTO_INCREMENT,
    nombre del certificado  VARCHAR(30),
    RESTRICCIÓN cer_id_pk CLAVE PRIMARIA (certID)
);

CREAR TABLA EmpCert (
    ID de empleado  ENTERO,
    certificadoID   ENTERO,
    RESTRICCIÓN empcer_pk CLAVE PRIMARIA (ID de empleado, ID de certificado),
    RESTRICCIÓN emp_id_fk CLAVE EXTRANJERA (ID de empleado) REFERENCIAS Empleado (ID de empleado),
```

```
    RESTRICCIÓN cer_id_fk CLAVE EXTRANJERA (ID de certificado) REFERENCIAS Certificado (ID de certificado)
);
```

Archivo POM

Solución

```
<?xml versión="1.0" codificación="UTF-8"?>

<proyecto xmlns="http://maven.apache.org/POM/4.0.0"                xmlns: xsi="http://www.w3.org/2001/ xsi:
  esquemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
  <modelVersion>4.0.0</modelVersion>

  <groupId>ceed.ada</groupId>
  <artifactId>U3HIBClassicExample</artifactId>
  <versión>0.0.1-SNAPSHOT</versión>

  <name>U3HIBClassicExample</name>
  <!-- FIXME cámbielo al sitio web del proyecto --> <url>http://
  www.example.com</url>

  <propiedades>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </propiedades>

  <dependencias>
    <dependencia>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <versión>4.11</versión>
      <scope>prueba</scope>
```

```
        </dependencia>
        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java --> <dependencia>

            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <versión>8.0.33</versión>
        </dependencia>
        <!-- https://central.sonatype.com/artifact/org.hibernate.orm/hibernate-core/6.4.0 <dependencia>

            <groupId>org.hibernate.orm</groupId>
            <artifactId>hibernate-core</artifactId>
            <versión>6.4.0.Final</versión>
        </dependencia>
    </dependencias>
</proyecto>
```

ORM Hibernar clásico

Solución

certificado.hbm.xml

```
<?xml versión = "1.0" codificación = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//ES"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<mapeo-hibernación>
    <nombre de clase = "DOMINIO.Certificado" tabla = "Certificado">
        <meta atributo = "descripción de clase">
            Esta clase contiene el detalle del certificado.
        </meta>
        <id nombre = "iCertID" tipo = "int" columna = "certID">
            <generador clase="nativo"/>
        </id>
        <nombre de propiedad = "stCertName" columna = "certname" tipo = "cadena"/>
    </clase>
</hibernate-mapping>
```

---

empleado.hbm.xml

```
<?xml versión = "1.0" codificación = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//ES"
```

```
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<mapeo-hibernación>
    <nombre de clase = "DOMINIO.Empleado" tabla = "Empleado">
        <atributo meta = "descripción-clase">
            Esta clase contiene los detalles del empleado. </meta>
        <id
        nombre = "iEmpID" tipo = "int" columna = "empID">
            <generador clase="nativo"/>
        </id>
          <!-- https://www.tutorialspoint.com/hibernate/hibernate_many_to_many_mapping. <set nombre =
          "relCertificates" cascade="save-update" table="EmpCert">
            <columna clave = "ID de empleado"/>
            <columna de muchos a muchos = "certificadoID" clase="DOMINIO.Certificado"/>
        </conjunto>

        <nombre de propiedad = "stFirstName" columna = "firstname" tipo = "cadena"/>
        <nombre de propiedad = "apellido" columna = "apellido" tipo = "cadena"/>
        <nombre de propiedad = "dSalario" columna = "salario" tipo = "doble"/>
    </clase>
</hibernate-mapping>
```

---

hibernación.cfg.xml

```
<?xml versión="1.0" codificación="utf-8"?>
<!DOCTYPE configuración-hibernación PUBLIC
"-//Hibernar/Configuración de hibernación DTD 3.0//ES"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<configuración-hibernación>
    <fábrica de sesiones>
        < nombre de propiedad="hibernate.connection.url">jdbc:mysql://localhost:3306/DBCerti
        < nombre de propiedad="hibernate.connection.nombre de usuario">usuariomaven</property>
        < nombre de propiedad="hibernate.connection.contraseña">ada0486</property>
```

```
            <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect <property
            name="show_sql">false</property> <property
            name="format_sql">true</property> <property
            name="hbm2ddl. auto">actualización</property> <mapping
            Resource="employee.hbm.xml" /> <mapping
            Resource="certificate.hbm.xml" /> </session-
        factory> </hibernate-
configuration>
```

DATOS. CertificadoDAO

Solución

```
DATOS del paquete ;

importar java.util.List;
importar java.util.Iterator;

importar org.hibernate.HibernateException; importar
org.hibernate.Session; importar
org.hibernate.Transaction;

importar DOMINIO.*;
importar UTIL.*;

/**
 *  ============================================== ==========================
 * Capa de datos. Programa para gestionar operaciones CRUD a la BD. Certificado de mesa *@autor
 Abelardo Martínez. Basado y modificado de Sergio Badal.
 *  ============================================== ==========================
 */

Certificado de clase públicaDAO {

    /*
     *  ------------------
     * INSERTAR
     *  ------------------
     */
```

```
/* Método para CREAR un certificado en la base de datos */ public
Certificate addCertificate(String stCertName) {

    Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction txDB
    = null; //transacción de base de datos Certificado objCertificate
    = new Certificate(stCertName);

    intente
        { txDB = hibSession.beginTransaction(); //inicia la transacción //la operación
        de guardar está obsoleta, pero sigue funcionando en la última versión //https://stackoverflow.com
        questions/71211904/alternative-to-using-depre hibSession.persist(objCertificate); txDB.commit();
        finaliza la transacción System.out.println("*****
        Elemento agregado.\n"); } catch
        (HibernateException hibe) { if (txDB ! = null)
    txDB.rollback(); //algo salió mal, así que
        revertir
            hibe.printStackTrace();

    } finalmente {
        hibSession.close(); //cerrar sesión de hibernación

    } devolver objCertificado;
}

/*
 *   ------------------
 * SELECCIONAR
 *   ------------------
 */
/* Método para LEER todos los certificados */ public
void listCertificates() {

    Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction txDB
    = null; //transacción de base de datos

    intente
        { txDB = hibSession.beginTransaction(); //inicia la transacción
```

```
            //el antiguo método createQuery está obsoleto, pero sigue funcionando en la última versión //
            https://www.roseindia.net/hibernate/hibernate5/hibernate-5-query-deprec List<Certificate>
            listCertificates = hibSession.createQuery("FROM Certificar si (listCertificates.isEmpty())

                System.out.println("******** No se encontraron elementos");
            demás
                System.out.println("***** Iniciar listado...\n");

            for (Iterator<Certificado> itCertificate = listCertificates.iterator Certificado certificado =
                (Certificado) itCertificate.next(); System.out.print(" Nombre del certificado:
                                                    " + certificado.getstCertName() +
            }
            txDB.commit(); //finaliza la transacción
        } catch (HibernateException hbe) { if (txDB !
            = null) txDB.rollback(); //
                algo salió mal, así que revertir
            hbe.printStackTrace();
        } finalmente {
            hibSession.close(); //cerrar sesión de hibernación
        }
}

/*
  *   ------------------
  * ACTUALIZAR
  *   ------------------
  */
/* Método para ACTUALIZAR el nombre de un certificado
*/ public void updateCertificate(int iCertID, String stCertName) {

    Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction txDB
    = null; //transacción de base de datos

    intente
        { txDB = hibSession.beginTransaction(); //inicia la transacción Certificado
        objCertificate = (Certificado) hibSession.get(Certificado objCertificate.setstCertName(stCertName
```

```
            //el método de actualización está obsoleto, pero sigue funcionando en la última versión //
            https://stackoverflow.com/questions/71211904/alternative-to-using-depre //https://stackoverflow.co
            questions/74124232/why -es-el-método-guardar-sav- hibSession.merge(objCertificate);
            txDB.commit(); //finaliza la transacción
            System.out.println("***** Elemento
            actualizado.\n");
        } catch (HibernateException e) { if (txDB !
            = null) txDB.rollback(); //
                algo salió mal, así que revertir
            e.printStackTrace();
        } finalmente {
            hibSession.close(); //cerrar sesión de hibernación
        }
    }

    /*
     *   -----------------
     * BORRAR
     *   -----------------
     */
    /* Método para ELIMINAR un certificado de los registros */ public void
    deleteCertificate(int iCertID) {

        Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction txDB
        = null; //transacción de base de datos

        intente
            { txDB = hibSession.beginTransaction(); //inicia la transacción Certificado
            objCertificate = (Certificado) hibSession.get(Certificado //el método de eliminación está
            obsoleto, pero aún funciona en la última versión //https://stackoverflow.com/questions/
            71211904/alternative-to-using-depre hibSession.remove(objCertificate); txDB.commit(); //finaliza
            la transacción System.out.println("*****
            Elemento eliminado.\n"); } catch
            (HibernateException hibe) { if (txDB != null ) txDB.rollback(); //
        algo salió mal, así que revertir
```

```
            hibe.printStackTrace();
        } finalmente {
            hibSession.close(); //cerrar sesión de hibernación
        }
    }

    /* Método para ELIMINAR todos los
    registros */ public void deleteAllItems() {

        Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction
        txDB = null; //transacción de base de datos

        intente
            { txDB = hibSession.beginTransaction(); //inicia la transacción //el
            antiguo método createQuery está obsoleto, pero sigue funcionando en la versión más
            reciente //https://www.roseindia.net/hibernate/hibernate5/hibernate-5-query-deprec
            List<Certificate> listCertificate = hibSession. createQuery("DESDE Certificado si (!
            listCertificate.isEmpty())
                for (Iterator<Certificate> itCertificate = listCertificate.iterator Certificate objCertificate
                    = (Certificate) itCertificate.next //el método de eliminación está obsoleto,
                    pero aún funciona en la última versión //https://stackoverflow.com/questions/
                    71211904/alternative- to-usi hibSession.remove(objCertificate);

                }
            txDB.commit(); //finaliza la transacción
        } catch (HibernateException hibe) { if
            (txDB ! = null)
                txDB.rollback(); //algo salió mal, así que revertir
            hibe.printStackTrace();
        } finalmente {
            hibSession.close(); //cerrar sesión de hibernación
        }
    }

}
```

DATOS. EmpleadoDAO

Solución

```
DATOS del paquete ;

importar java.util.List;
importar java.util.Set;
importar java.util.Iterator;

importar org.hibernate.HibernateException; importar
org.hibernate.Session; importar
org.hibernate.Transaction;

importar DOMINIO.*;
importar UTIL.*;

/**
 *  ============================================= ==========================
 * Capa de datos. Programa para gestionar operaciones CRUD a la BD. Empleado de mesa *
 @autor Abelardo Martínez. Basado y modificado de Sergio Badal.
 *  ============================================= ==========================
 */

clase pública EmpleadoDAO {

    /*
     *  ------------------
     * INSERTAR
     *  ------------------
```

```
 */
/* Método para CREAR un empleado en la base de datos */ public
Employee addEmployee(String stFirstName, String stLastName, double dSalary

    Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction txDB
    = null; //transacción de base de datos Empleado objEmployee
    = new Empleado(stFirstName, stLastName, dSalary);

    intente
        { txDB = hibSession.beginTransaction(); //inicia la transacción
        objEmployee.setrelCertificates(setCertificates);
        //el método guardar está obsoleto, pero sigue funcionando en la última versión //https://
        stackoverflow.com/questions/71211904/alternative-to-using-depre hibSession.persist(objEmploye
        txDB.commit(); //finaliza la transacción
        System.out.println("***** Elemento agregado.
        \n"); } catch (HibernateException hibe) { if (txDB ! = null)
    txDB.rollback(); //algo salió mal, así que
        revertir
            hibe.printStackTrace();

    } finalmente {
        hibSession.close(); //cerrar sesión de hibernación

    } return objEmpleado;
}

/*
 *   ------------------
 * SELECCIONAR
 *   ------------------
 */
/* Método para LEER todos los empleados */
public void listEmployees() {

    Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction txDB
    = null; //transacción de base de datos
```

```
        intentar {
            txDB = hibSession.beginTransaction(); //inicia la transacción
            //el antiguo método createQuery está obsoleto, pero sigue funcionando en la última versión //
            https://www.roseindia.net/hibernate/hibernate5/hibernate-5-query-deprec List<Employee>
            listEmployees = hibSession.createQuery("FROM Empleado" si (listEmployees.isEmpty())

                System.out.println("******** No se encontraron elementos");
            demás
                System.out.println("\n***** Iniciar listado...\n");

            for (Iterador<Empleado> itEmpleado = listaEmpleados.iterator(); itEmpleado
                Empleado objEmpleado = (Empleado) itEmployee.next();
                System.out.print(" Nombre: + objEmployee.getstFirstName()
                                                    "
                System.out.print("Apellido : + objEmployee.getstLastName() +
                                                    "
                System.out.println("Salario: " + objEmployee.getdSalary()); Set<Certificado>
                setCertificates = objEmployee.getrelCertificates for (Iterator<Certificado> itCertificate
                = setCertificates.iterator Certificado objCertificate = (Certificado) itCertificate.next
                    System.out.println("Certificado: " + objCertificate.getstCertName

                }
            }
            txDB.commit(); //finaliza la transacción
        } catch (HibernateException hibe) { if (txDB ! =
            null) txDB.rollback(); //
                algo salió mal, así que revertir hibe.printStackTrace();

        } finalmente {
            hibSession.close(); //cerrar sesión de hibernación
        }
    }

    /*
     *   ------------------
     * ACTUALIZAR
     *   ------------------
     */
    /* Método para ACTUALIZAR el salario de un empleado */
```

```
public void updateEmployee(int iEmpID, doble dSalario) {

    Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction txDB
    = null; //transacción de base de datos

    intente
        { txDB = hibSession.beginTransaction(); // inicia la transacción Empleado
        objEmployee = (Employee) hibSession.get(Employee.class, iEmpID
        objEmployee.setdSalary(dSalary); //el
        método de actualización está obsoleto, pero aún funciona en la última versión //https://
        stackoverflow.com/ questions/71211904/alternative-to-using-depre hibSession.merge(objEmploye
        txDB.commit(); //finaliza la transacción
        System.out.println("***** Artículo actualizado.
        \n"); } catch (HibernateException hibe) { if (txDB != null)
    txDB.rollback(); //algo salió mal, así que
        revertir
            hibe.printStackTrace();

    } finalmente {
        hibSession.close(); //cerrar sesión de hibernación
    }
}

/*
 *   ----------
 * BORRAR
 *   ----------
 */
/* Método para ELIMINAR un empleado de los registros */ public
void deleteEmployee(int iEmpID) {

    Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction txDB
    = null; //transacción de base de datos

    intente
        { txDB = hibSession.beginTransaction(); //inicia la transacción Empleado
        objEmpleado = (Empleado) hibSession.get(Empleado.clase, iEmpID
```

```
            //el método de eliminación está obsoleto, pero sigue funcionando en la última versión //
            https://stackoverflow.com/questions/71211904/alternative-to-using-depre
            hibSession.remove(objEmployee);
            txDB.commit(); //finaliza la transacción
            System.out.println("***** Elemento eliminado.\n"); }
        catch (HibernateException hibe) { if (txDB !
            = null) txDB.rollback(); //
                algo salió mal, así que revertir hibe.printStackTrace();

        } finalmente {
            hibSession.close(); //cerrar sesión de hibernación
        }
    }

    /* Método para ELIMINAR todos los
    registros */ public void deleteAllItems() {

        Sesión hibSession = HibernateUtil.SFACTORY.openSession(); //abrir hibernación s Transaction
        txDB = null; //transacción de base de datos

        intentar {
            txDB = hibSession.beginTransaction(); //inicia la transacción
            //el antiguo método createQuery está obsoleto, pero sigue funcionando en la última
            versión //https://www.roseindia.net/hibernate/hibernate5/hibernate-5-query-deprec
            List<Employee> listEmployees = hibSession.createQuery("FROM Empleado" si (!
            listEmployees.isEmpty())
                for (Iterador<Empleado> itEmpleado = listEmployees.iterator();
                    Empleado objEmpleado = (Empleado) itEmployee.next(); //el
                    método de eliminación está obsoleto, pero sigue funcionando en la última
                    versión //https://stackoverflow.com/questions/71211904/alternative-to-usi
                    hibSession.remove(objEmployee);
                }
            txDB.commit(); //finaliza la transacción
        } catch (HibernateException hibe) { if
            (txDB ! = null)
                txDB.rollback(); //algo salió mal, así que revertir
            hibe.printStackTrace();
```

```
            } finalmente {
                hibSession.close(); //cerrar sesión de hibernación
            }
        }

    }
```

DOMINIO. Certificado

Solución

```
paquete DOMINIO;

/**
 *   ============================================== ==================
 * Certificado de Objeto
 *@autor Abelardo Martínez. Basado y modificado de Sergio Badal.
 *   ============================================== ==================
 */

Certificado de clase pública {

    /*
     *   ----------------------------------------
     * ATRIBUTOS
     *   ----------------------------------------
     */
    /*
     * Tenga cuidado con el nombre de las variables. Los captadores y definidores deben seguir * para
     encontrar los métodos apropiados
     * https://stackoverflow.com/questions/921239/hibernate-propertynotfoundexception
     */
    privado int iCertID;
    cadena privada stCertName;

    /*
     *   ----------------------------------------
```

```
 * MÉTODOS
 *    ----------------------------------------
 */ /
*
 * Constructor vacío */

Certificado público () { } /* *


Constructor sin ID. Todos los campos, excepto la clave principal */

Certificado público (String stCertName) {
        this.stCertName = stCertName;
}

/*
 *    ----------------------------------------
 * RECOGEDORES
 *    ----------------------------------------
 */
público int getiCertID() { return
        iCertID;
}

cadena pública getstCertName() {
        devolver stCertName;
}

/*
 *    ----------------------------------------
 * CONFIGURADORES
 *    ----------------------------------------
 */
public void setiCertID(int iCertID) { this.iCertID =
        iCertID;
}
```

```
    setstCertName público vacío (String stCertName) {
        this.stCertName = stCertName;
    }

    /*
     * Establecer la relación MUCHOS A MUCHOS entre Empleado y Certificado
     *Elegimos el lado derecho (mesa Certificado)
     * Crear métodos iguales () y hashCode () para que Java pueda determinar si hay alguno */

    público booleano es igual (Objeto obj) {
        si (obj == nulo)
            falso retorno ;
        si (!this.getClass().equals(obj.getClass())) devuelve
            falso;

        Certificado objCert = (Certificado) obj; if
        ((this.iCertID == objCert.getiCertID()) && (this.stCertName.equals(
            devolver verdadero;

        } falso retorno ;
    }

    público int hashCode() { int
        iHash = 0; iHash
        = (iCertID + stCertName).hashCode(); devolver
        iHash;
    }

}
```

DOMINIO. Empleado

Solución

```
paquete DOMINIO;

importar java.util.Set;

/**
 *  ============================================== ==================
 *Objeto Empleado*
 @autor Abelardo Martínez. Basado y modificado de Sergio Badal.
 *  ============================================== ==================
 */

Empleado de clase pública {

    /*
     *  ---------------------------------------
     * ATRIBUTOS
     *  ---------------------------------------
     */ /
    *
     * Ojo con el nombre de las variables. Los captadores y definidores deben seguir * para encontrar los
     métodos apropiados * https://
     stackoverflow.com/questions/921239/hibernate-propertynotfoundexception */

    privado int iEmpID;
    cadena privada stFirstName;
    cadena privada apellido;
```

```
doble salario privado ; //
Establecer clase en
Java //https://www.geeksforgeeks.org/set-in-java/ private
Set<Certificate> relCertificates; //relación Empleado-Certificado (N:

/*
  *   ----------------------------------------
  * MÉTODOS
  *   ----------------------------------------
  */ /
*
  * Constructor vacío
  */
public Employee() { } /* *


Constructor sin ID. Todos los campos, excepto la clave principal */

Empleado público (String stFirstName, String stLastName, doble dSalario) {
        this.stFirstName = stFirstName;
        this.stLastName = stLastName;
        this.dSalario = dSalario;
}

/*
  *   ----------------------------------------
  * RECOGEDORES
  *   ----------------------------------------
  */
public int getiEmpID() { return
        iEmpID;
}

cadena pública consigue primer nombre() {
        devolver stFirstName;
}
```

```
cadena pública obtieneApellido() {
        devolver apellido;
}

público doble getdSalario() {
        devolver dSalario;
}

conjunto público <Certificado> getrelCertificados() {
        devolver certificados rel;
}

/*
  *    ----------------------------------------
  * CONFIGURADORES
  *    ----------------------------------------
  */
public void setiEmpID(int iempID) { this.iEmpID
        = iempID;
}

setstFirstName público vacío (String stFirstName) {
        this.stFirstName = stFirstName;
}

public void setstLastName(String stLastName) { this.stLastName
        = stLastName;
}

public void setdSalary(doble dSalario) { this.dSalario =
        dSalario;
}

public void setrelCertificates(Set<Certificado> relCertificates) { this.relCertificates =
        relCertificates;
}
```

```
}
```

INTERFAZ. PruebaHibernarMySQL

Solución

```
paquete INTERFAZ;

importar java.util.HashSet;

datos de
importacion .*; importar
DOMINIO.*; importar UTIL.*;

/**
 *   ============================================= ==================
 * Capa de interfaz. Programa para gestionar PERSONAS
 *@autor Abelardo Martínez. Basado y modificado de Sergio Badal.
 *   ============================================= ==================
 */

/*
 * Para más información ver este tutorial:
 * https://www.tutorialspoint.com/hibernate/hibernate_examples.htm */


clase pública TestHibernateMySQL {

        /*
         *   -----------------
         * PROGRAMA PRINCIPAL
         *   -----------------
```

```
 */
principal vacío estático público (cadena [] stArgs) {

    //Crear nuevos objetos DAO para operaciones CRUD
    EmpleadoDAO objEmpleadoDAO = nuevo EmpleadoDAO();
    CertificateDAO objCertificateDAO = nuevo CertificateDAO();

    //TRUNCAR TABLAS. Eliminar todos los registros de las tablas
    objEmployeeDAO.deleteAllItems();
    objCertificateDAO.deleteAllItems();

    /* Agregar registros en la base de datos */
    Certificado objCert1 = objCertificateDAO.addCertificate("MBA"); Certificado
    objCert2 = objCertificateDAO.addCertificate("PMP");

    //Conjunto de certificados
    HashSet<Certificate> hsetCertificates = new HashSet<Certificate>();
    hsetCertificates.add(objCert1);
    hsetCertificates.add(objCert2);

    /* Agregar registros en la base de datos */
    Empleado objEmp1 = objEmployeeDAO.addEmployee("Alfred", "Vincent", 4000 Empleado
    objEmp2 = objEmployeeDAO.addEmployee("John", "Gordon", 3000, hsetCer

    /* Actualizar el campo de salario del empleado
    */ objEmployeeDAO.updateEmployee(objEmp1.getiEmpID(), 5000); /*
    Enumera todos los empleados */
    objEmployeeDAO.listEmployees(); /*
    Eliminar un empleado de la base de datos */
    objEmployeeDAO.deleteEmployee(objEmp2.getiEmpID()); /*
    Enumere todos los certificados */
    objCertificateDAO.listCertificates(); /* Enumera
    todos los empleados */
    objEmployeeDAO.listEmployees();

    //Cerrar la fábrica de sesiones de hibernación
    global HibernateUtil.shutdownSessionFactory();
```

```
        }

    }
```

UTIL. HibernateUtil

Solución

```
paquete UTIL;

importar java.util.logging.Level;

importar org.hibernate.SessionFactory; importar
org.hibernate.cfg.Configuration;

/**
 *   ================================================ ==================
 *Clase para gestionar sesión de Hibernate*
 @autor Abelardo Martínez. Basado y modificado de Sergio Badal.
 *   ================================================ ==================
 */

clase pública HibernateUtil {

    /**
     *   ---------------------------------------
     * CONSTANTES Y VARIABLES GLOBALES
     *   ---------------------------------------
     */
    // Sesión persistente public
    static final SessionFactory SFACTORY = buildSessionFactory();

    /*
     *   ---------------------
```

```
     * GESTIÓN DE SESIONES
     *    ----------------------
     */ /
    *
     * Crear nueva sesión de hibernación
     */
    SessionFactory estática privada buildSessionFactory() {
        java.util.logging.Logger.getLogger("org.hibernate").setLevel(Level.OFF intente {

            // Crea SessionFactory desde hibernate.cfg.xml return new
            Configuration().configure().buildSessionFactory();
        } catch (espejo lanzable) {
            // Asegúrese de registrar la excepción, ya que podría ser tragada
            System.err.println(" Falló la creación de SessionFactory." + sfe); lanzar un
            nuevo ExceptionInInitializerError(sfe);
        }
    }

    /*
     * Cerrar sesión de hibernación
     */
    ShutdownSessionFactory público estático vacío () {
        // Cerrar cachés y grupos de conexiones
        getSessionFactory().close();
    }

    /*
     *Obtener método para obtener la sesión
     */
    public static SessionFactory getSessionFactory() {
        devolver FÁBRICA;
    }

}
```