



Hibernate en Eclipse

David Cristian Pirosca

**Índice**

[**Proyecto Hibernate en Eclipse** 1](#_Toc154407231)

[**Base de Datos empresa.** 1](#_Toc154407232)

[**Instalamos Plugin JBoss Tools del Matketplace.** 1](#_Toc154407233)

[**Creamos un nuevo Proyecto Maven.** 3](#_Toc154407234)

[**Descargamos el Jar del Conector y Creamos una Conectividad con la Base de Datos.** 5](#_Toc154407235)

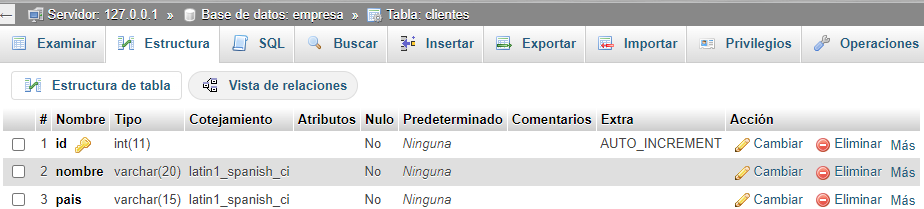
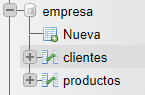
[**Creamos los archivos de Hibernate para poder hacer un mapeo de objetos** 10](#_Toc154407236)

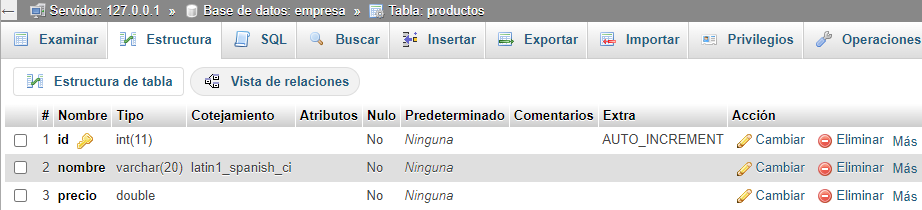
[**Buscamos las dependencias de Hibernate ORM Hibernate Core última versión.** 25](#_Toc154407237)

[**Usar Hibernate para hacer operaciones en la Base de Datos.** 27](#_Toc154407238)

# **Proyecto Hibernate en Eclipse**

## **Base de Datos empresa.**

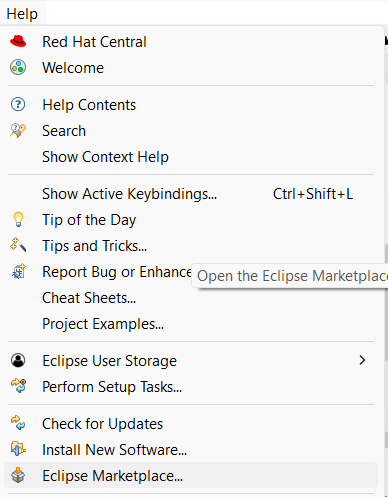
****

****

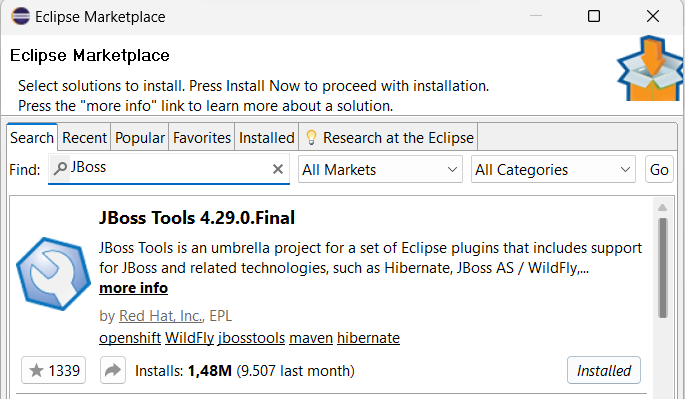
**Descargar empresa.sql**

[**https://github.com/davidpirosca/davidpirosca.github.io/blob/main/Codigos/Java/archivos/empresa.sql**](https://github.com/davidpirosca/davidpirosca.github.io/blob/main/Codigos/Java/archivos/empresa.sql)

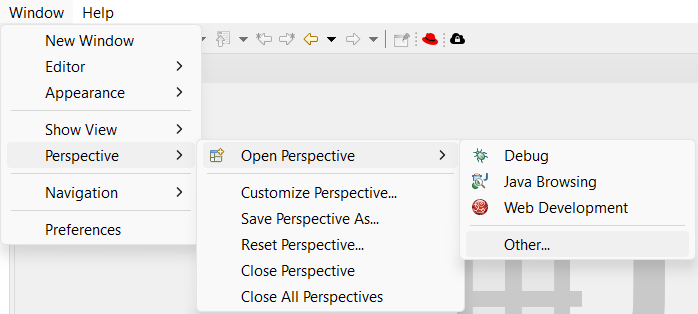
## **Instalamos Plugin JBoss Tools del Matketplace.**

****

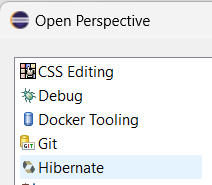
**Help > Eclipse Marketplace**

****

**Buscamos JBoss y lo Instalamos con la configuración por Defecto.**

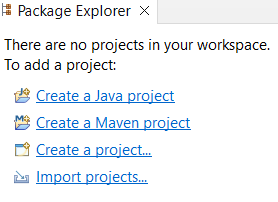
****

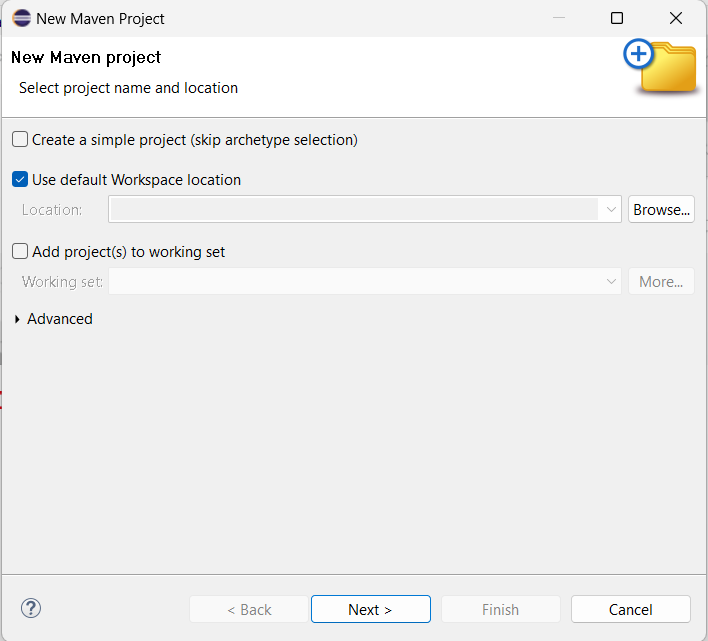
**Comprobamos que está instalado Hibernate en Window > Perspective > Open Perspective > Other…**

****

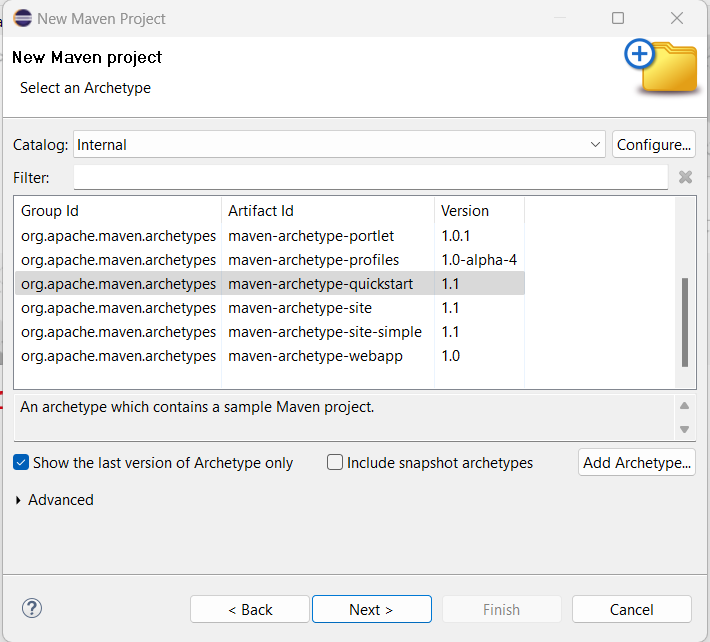
**Podremos observar que está instalado.**

## **Creamos un nuevo Proyecto Maven.**

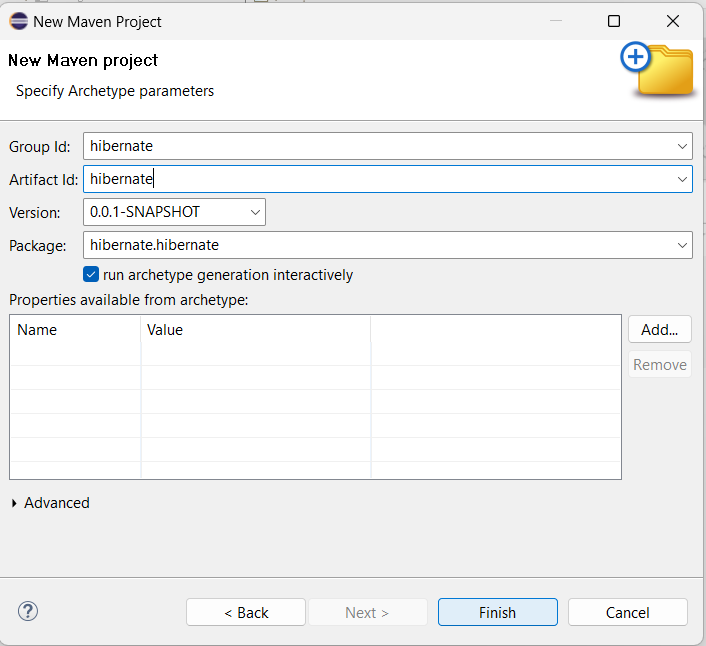
****

****

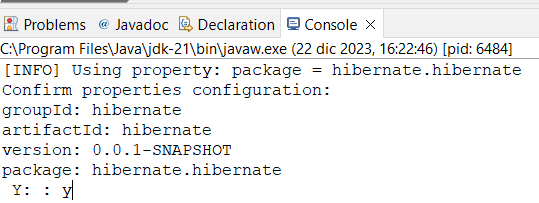
**En este caso lo guardamos en la ubicación del Espacio de Trabajo si quisiéramos otra ubicación lo podríamos cambiar y le damos a next.**

****

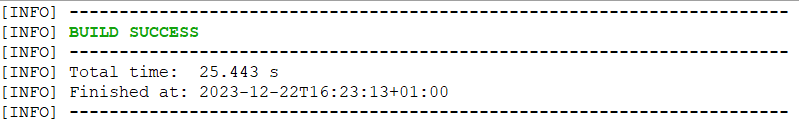
**En Catalog seleccionamos Internal y después buscamos la opción “maven-archetype-quickstart”.**

****

**Le ponemos nombre y le damos a finish.**

****

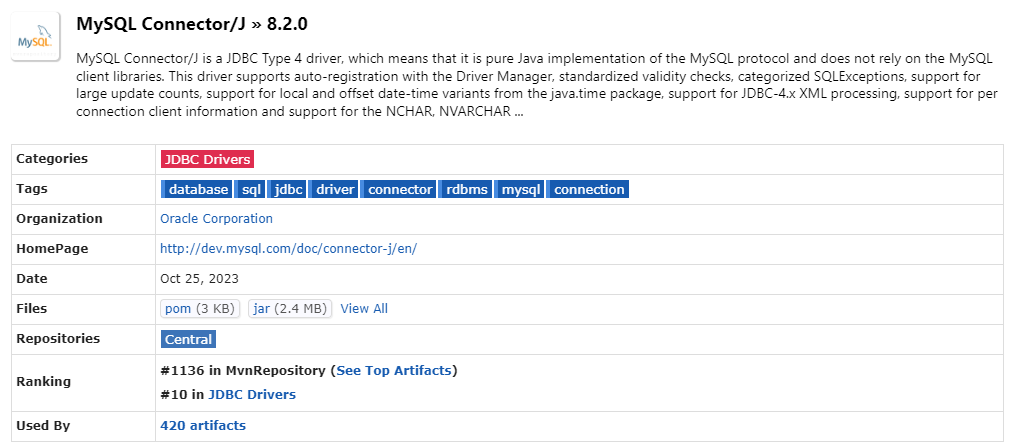
**En la consola nos piden confirmación para la creación ponemos “y” para poder seguir.**

****

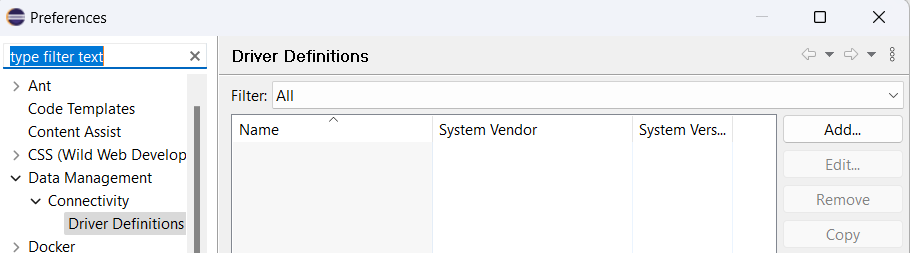
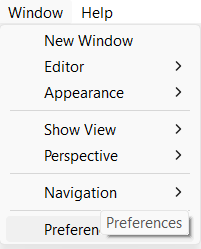
**En el caso de que se ha creado correctamente el proyecto nos aparecerá este mensaje.**

## **Descargamos el Jar del Conector y Creamos una Conectividad con la Base de Datos.**

[**https://mvnrepository.com/artifact/com.mysql/mysql-connector-j**](https://mvnrepository.com/artifact/com.mysql/mysql-connector-j)

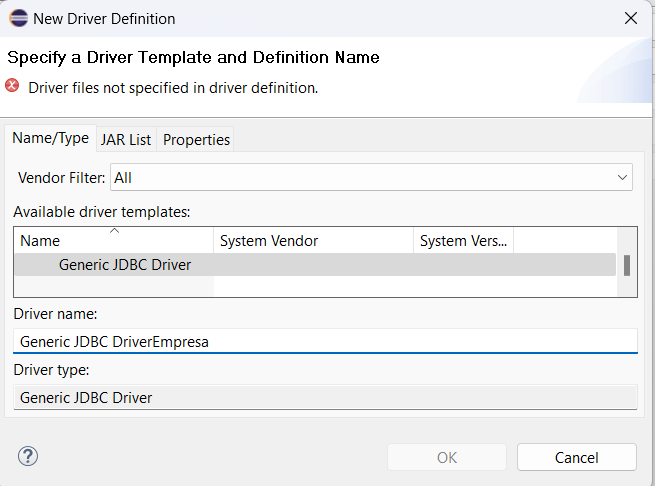
****

**Nos descargamos el Jar de la última versión.**

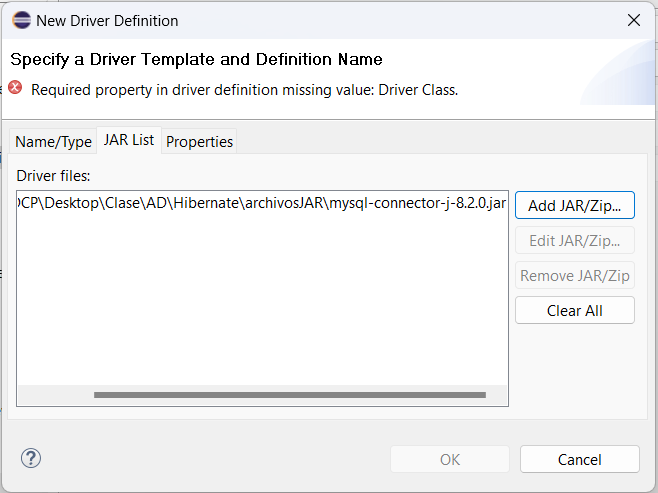
****

**Vamos a Window > Preference > Data Management > Conectivity > Add..**

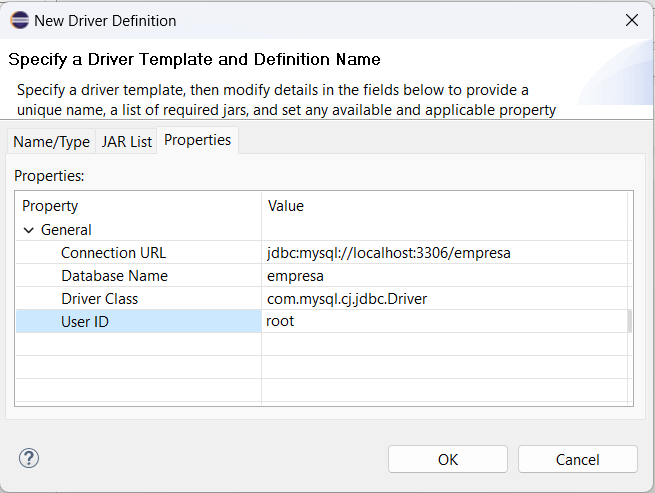
**Para agregar una nueva conectividad a la base de datos.**

****

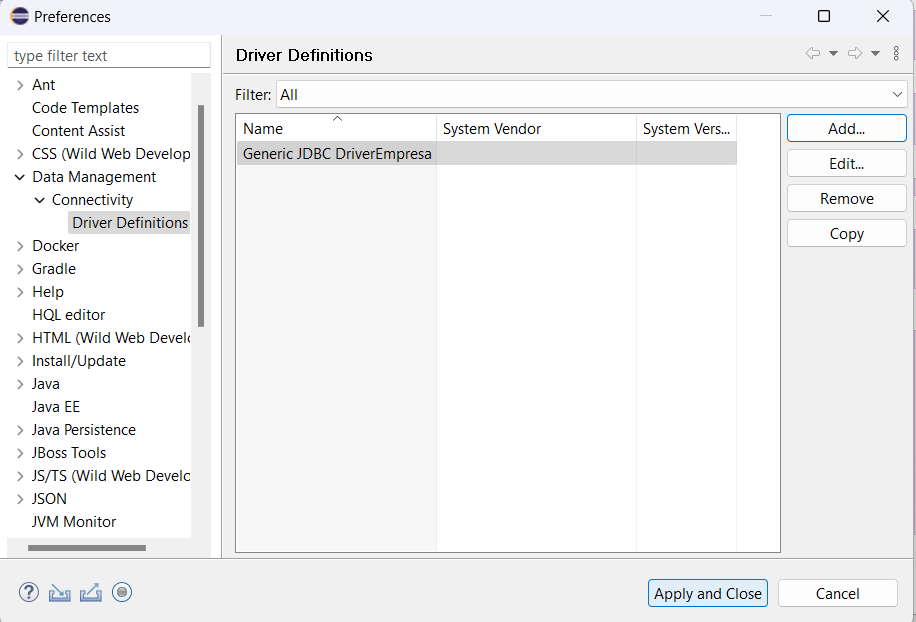
**Seleccionamos el Driver y le ponemos el nombre que queramos en este caso se le ha puesto Empresa para saber que la conexión se hará sobre la base de datos empresa.**

****

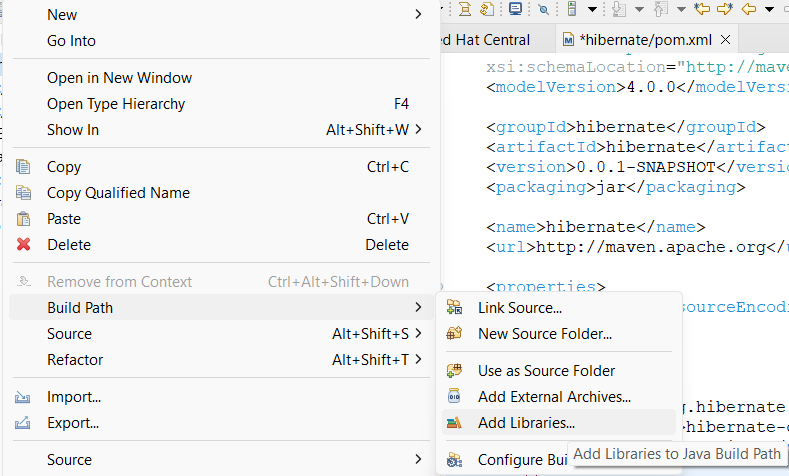
**En la siguiente pestaña tendremos que poner la JAR del conector que acabamos de descargarnos.**

****

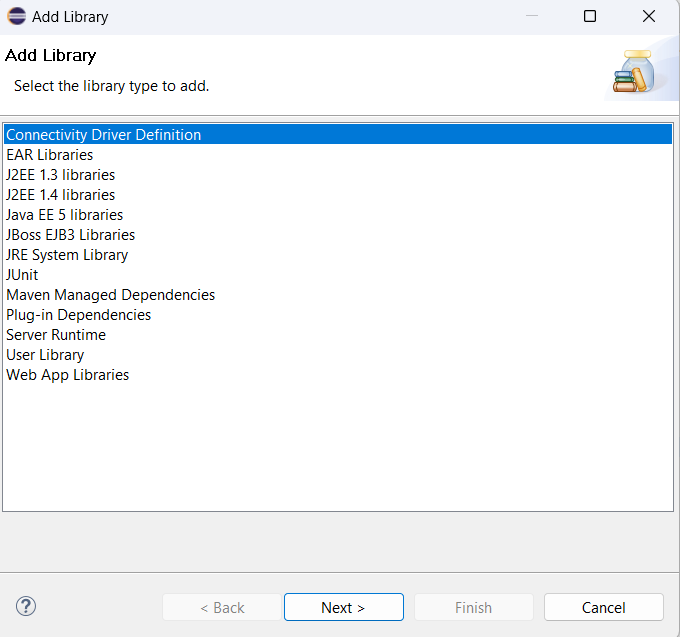
**En la siguiente pestaña tendremos que poner los datos de la base de datos que nos vamos a conectar. Y el Driver que este caso es el más nuevo.**

****

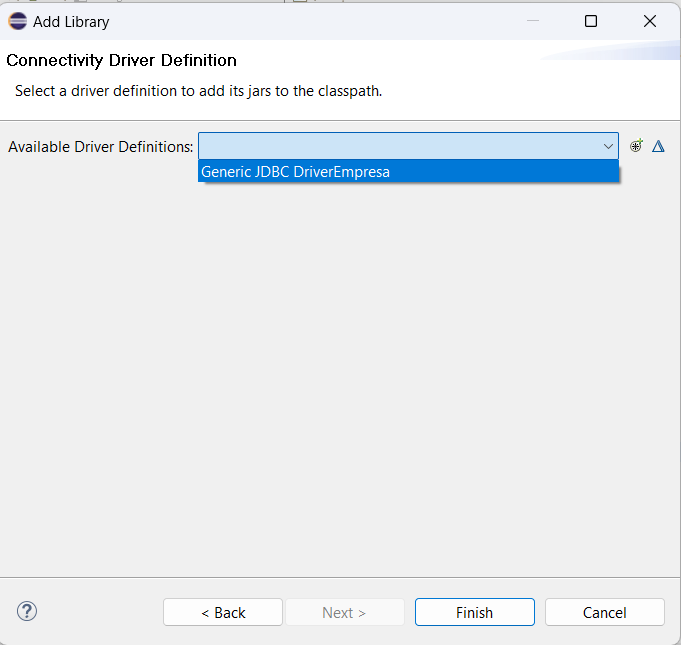
**Aceptamos y le damos a aplicar y cerrar.**

****

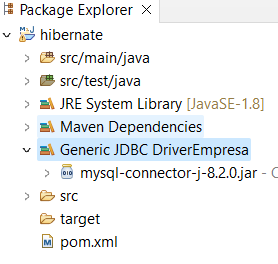
**Hacemos clic derecho sobre el proyecto > Build Path > Add Libraries…**

****

**Elegimos la primera opción que es Connectivity Driver Definition que es lo que acabamos de crear y le damos a Next.**

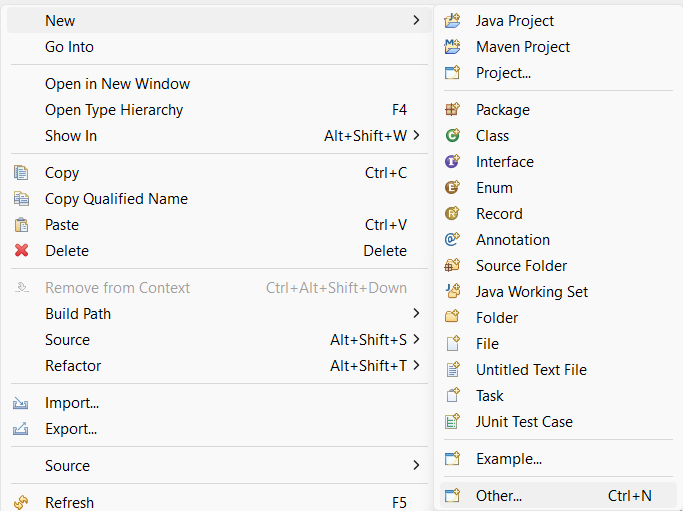
****

**En la siguiente ventana le damos al desplegable y elegimos lo que acabamos de crear que nos aparece, y por último le damos a Finish.**

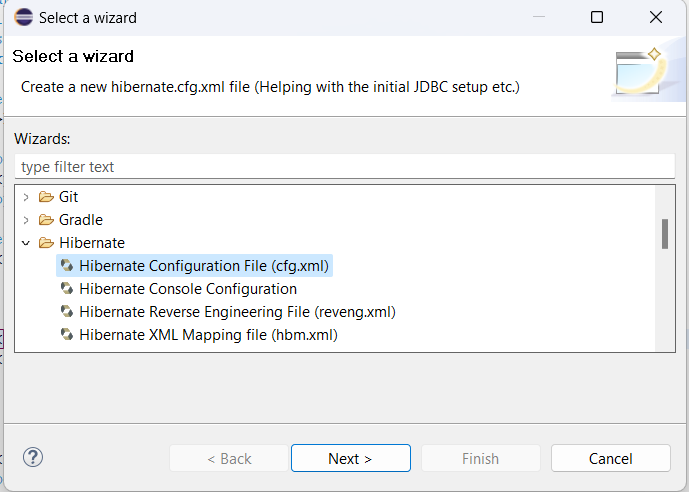
****

**Como podremos observar en el proyecto ya lo tenemos.**

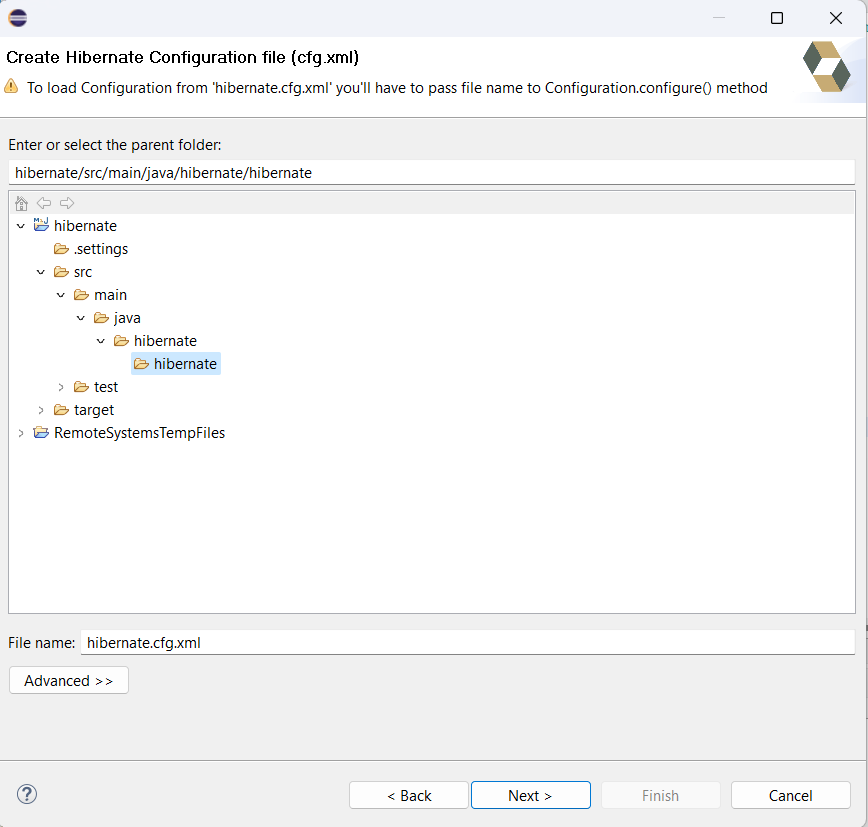
## **Creamos los archivos de Hibernate para poder hacer un mapeo de objetos**

****

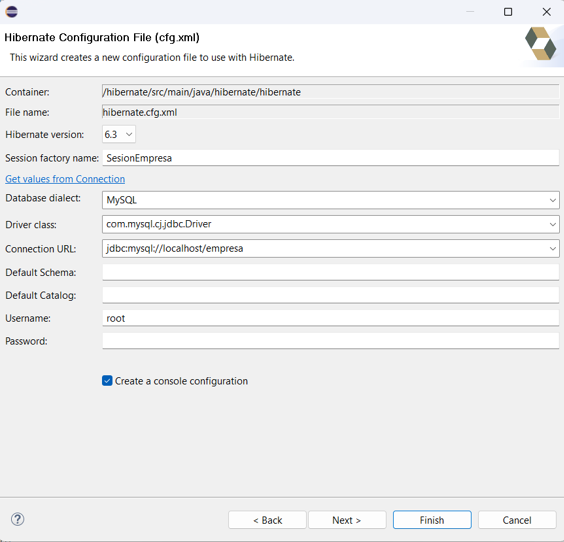
**Ahora vamos a crear y configurar el archivo de Configuración de Hibernate. Que esta en New > Other…**

****

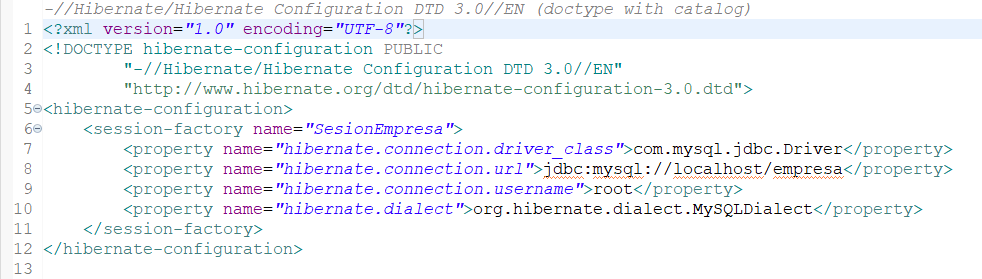
**Buscamos la carpeta de Hibernate y dentro tendremos Hibernate Configuration File (cfg.xml) y le damos a Next.**

****

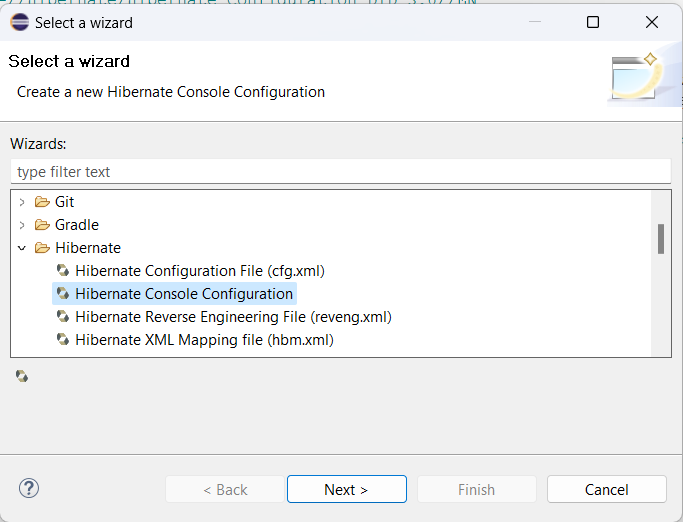
**Seleccionamos la carpeta donde queremos que se guarde el archivo y un nombre identificativo para el archivo y le damos a Next.**

****

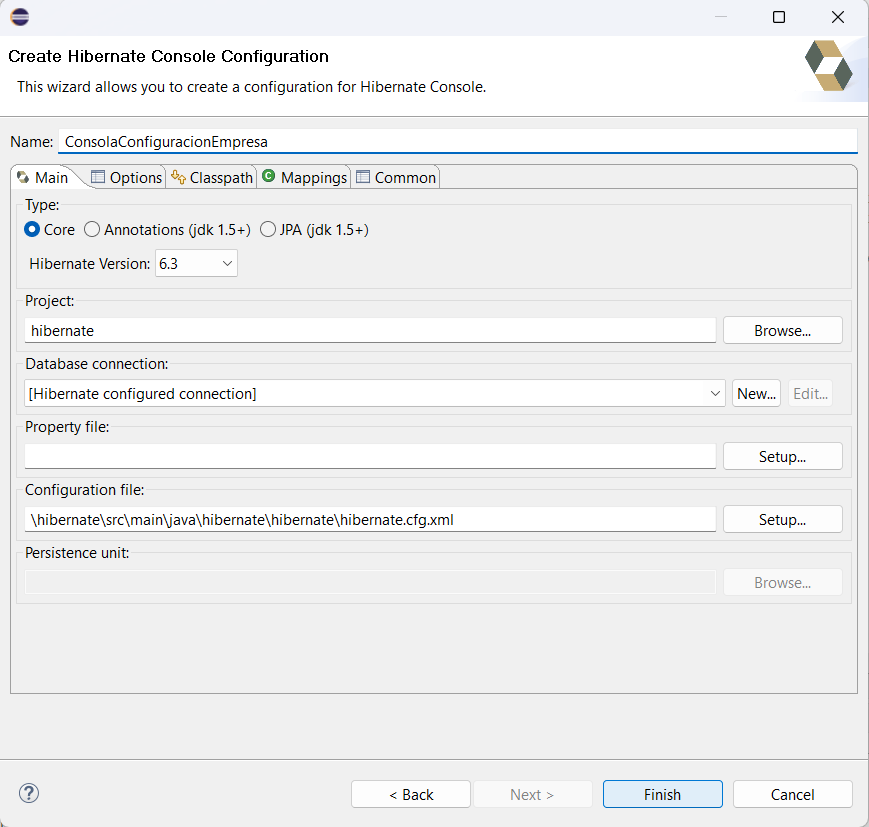
**Rellenamos con la configuración de la base de datos y le ponemos un nombre y le damos al Check de Create a console configuration. Por último, le damos a Finish.**

****

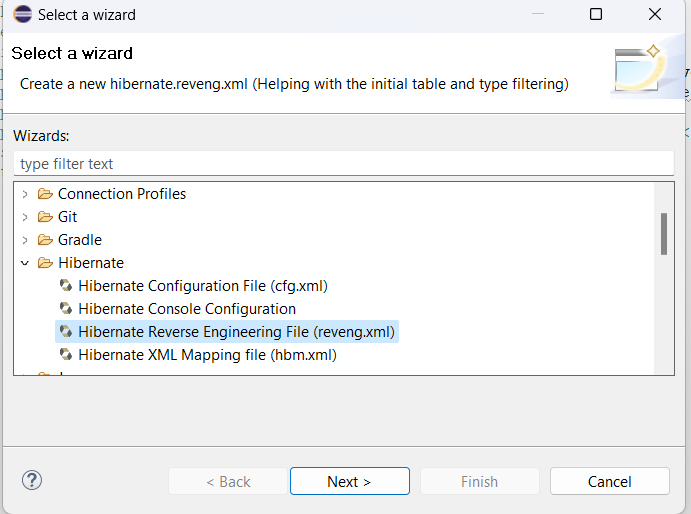
**Como podemos observar ya se nos ha creado la sesión con nuestros datos de conexión.**

****

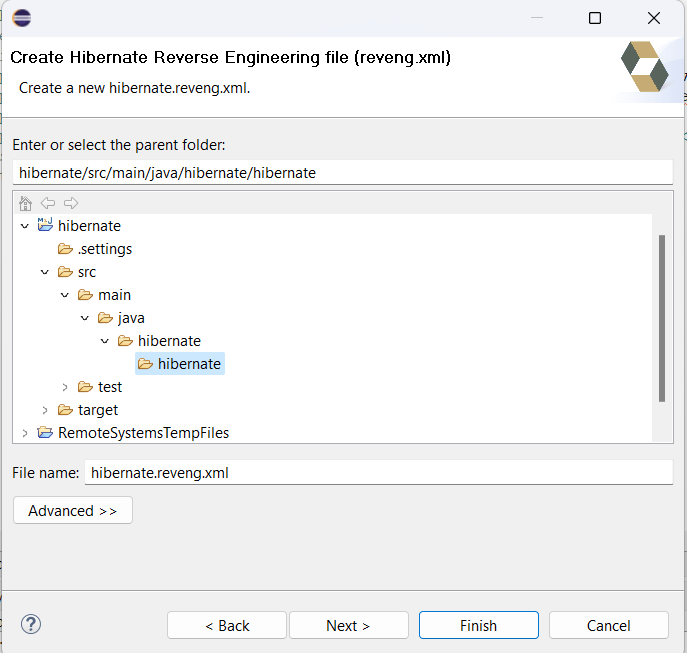
**Lo siguiente que tenemos que hacer es crear otro fichero, pero en este caso será de Hibernate Console Configuration.**

****

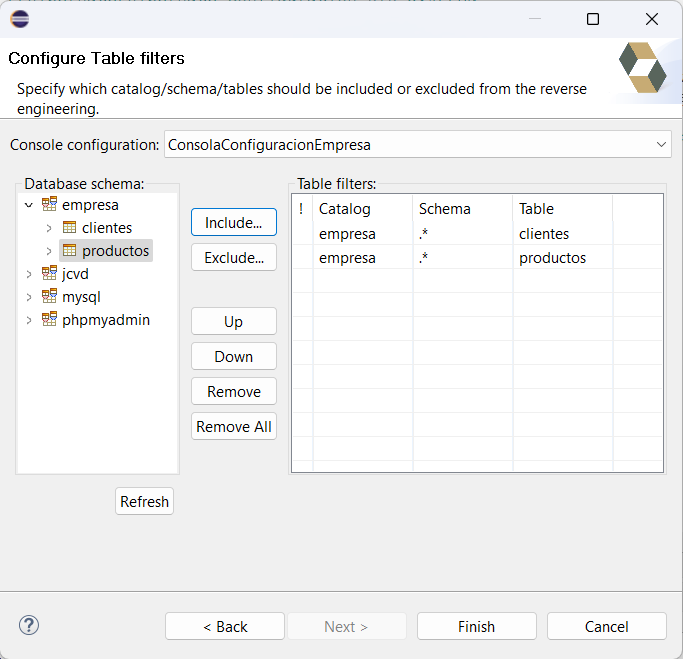
**Le ponemos un nombre y en caso de que faltase algún dato lo rellenamos en este caso el archivo de configuración y el proyecto los ha puesto el directamente por el paso anterior. Le damos por último a Finish. No muestra ningún Mensaje, pero la consola se crea correctamente.**

****

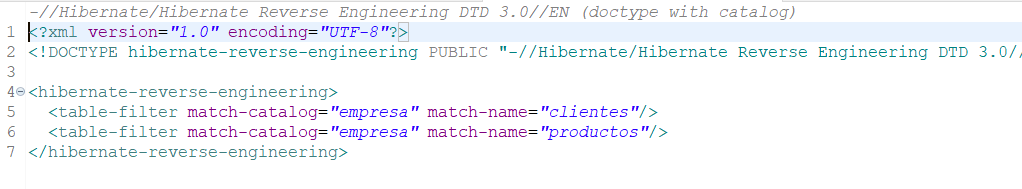
**Creamos otro archivo en este caso será Hibernate Reverse Engineering File (reveng.xml).**

****

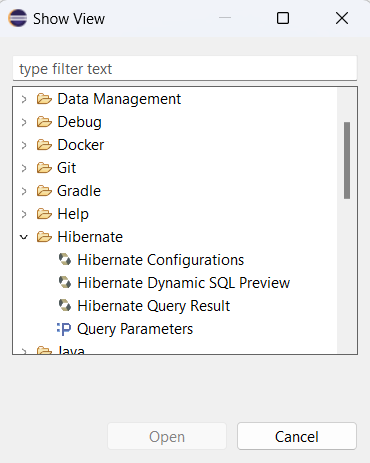
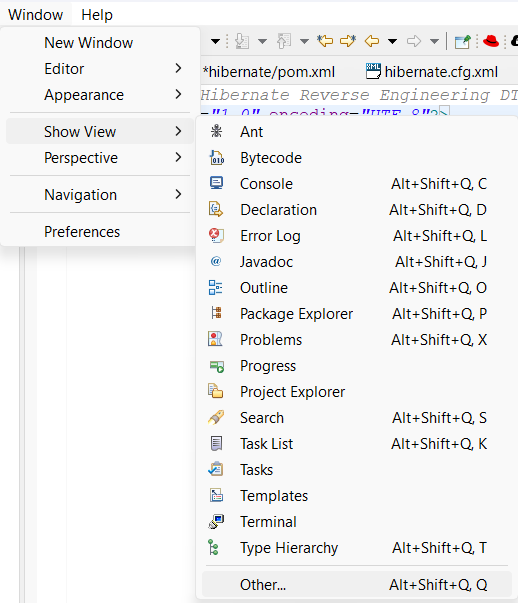
**Elegimos la carpeta donde hemos guardado anteriormente el hibernate.cfg.xml.**

****

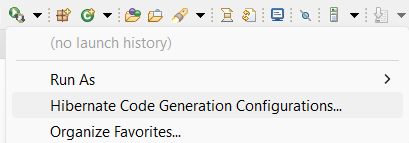
**Elegimos del desplegable la consola que hemos creado y le damos a Refresh y podremos observar que se conecta a la base de datos y podremos ver todas las tablas a las que tengamos acceso en este caso con el usuario root tenemos acceso a todas elegimos la que nos interesa que en nuestro caso es empres y elegimos las dos tablas y le damos a Include para que los incluya en los filtros y por últimos le damos a Finish.**

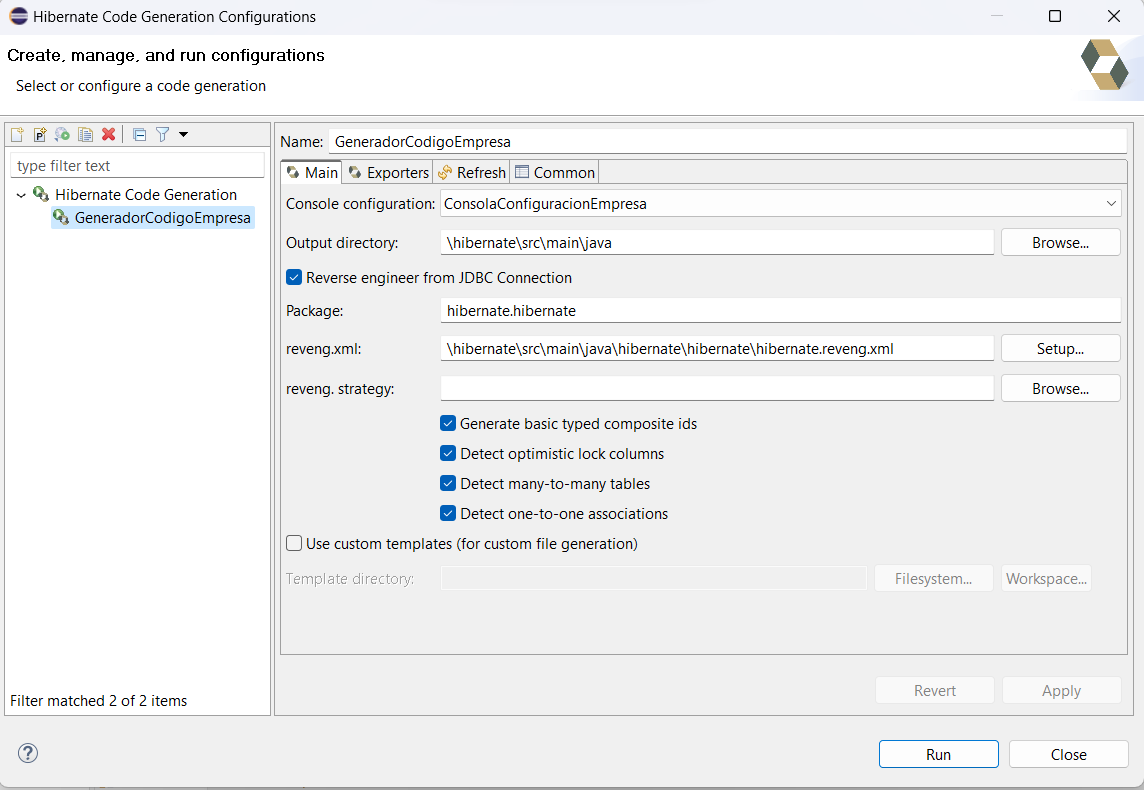
****

**Se nos crea el fichero con los parámetros que le hemos metido antes.**

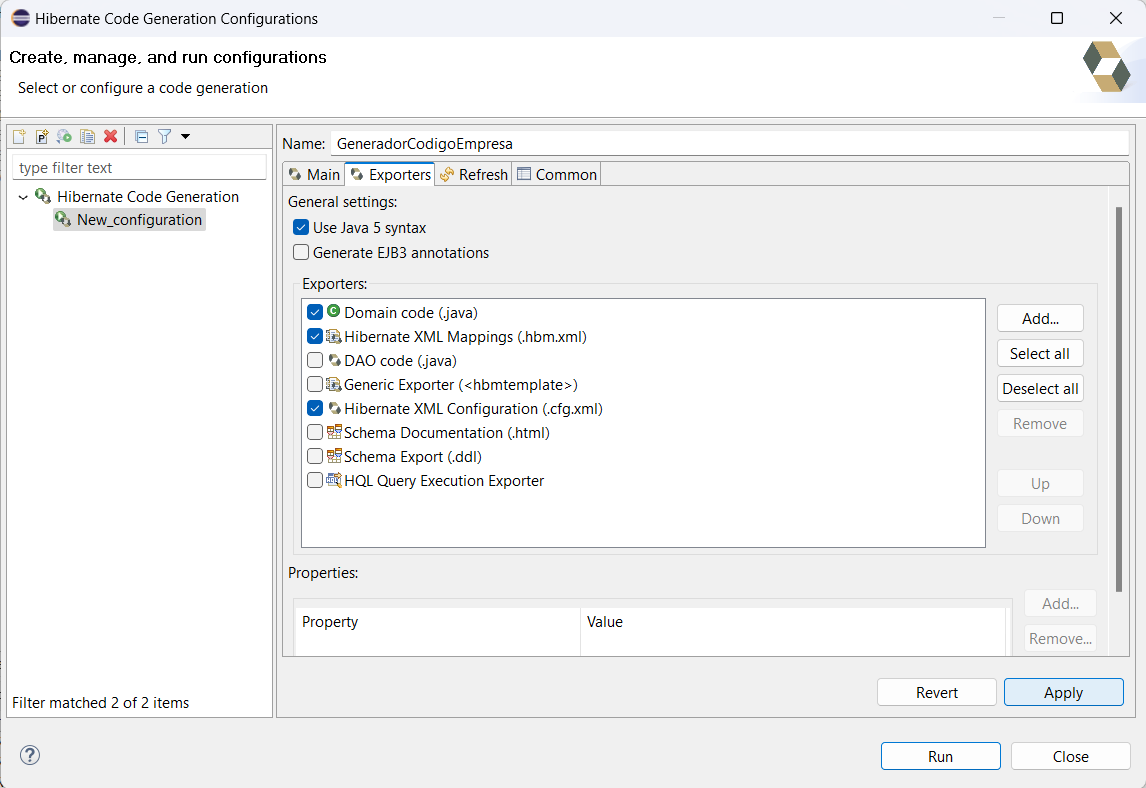
****

**Window > Show View > Other > Hibernate > Hibernate Configurations**

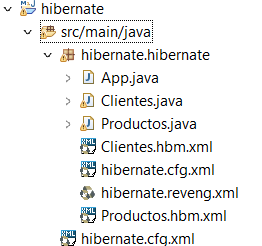
****

****

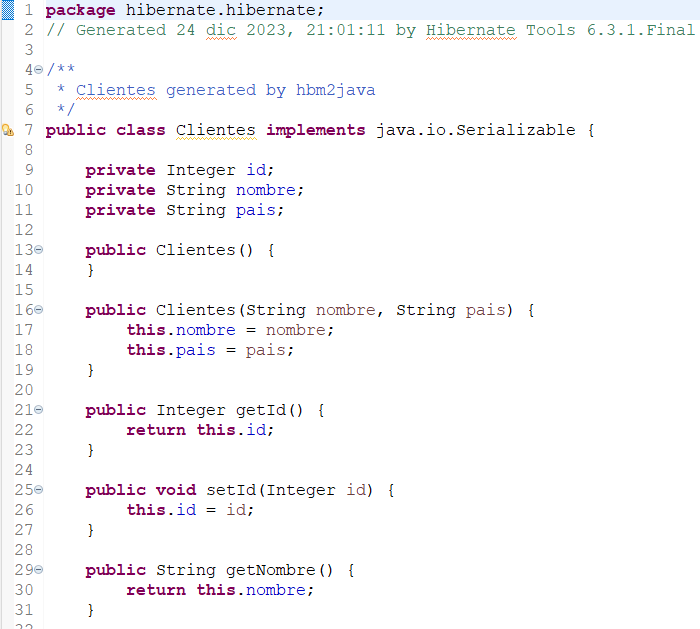
**Se nos abrirá la siguiente ventana en la cual tendremos que poner los datos de la configuración de consola, directorio , paquete en el que creará las clases y el archivo de reveng.xml que contendrá la información sobre las tablas. Vamos a la siguiente pestaña para configurar lo siguiente.**

****

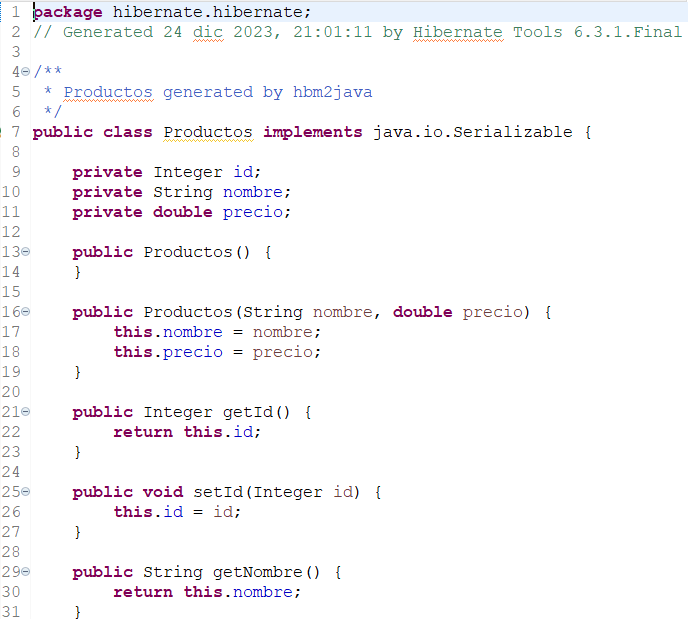
**Marcamos las opciones de la imagen y después le damos a Apply y Run.**

****

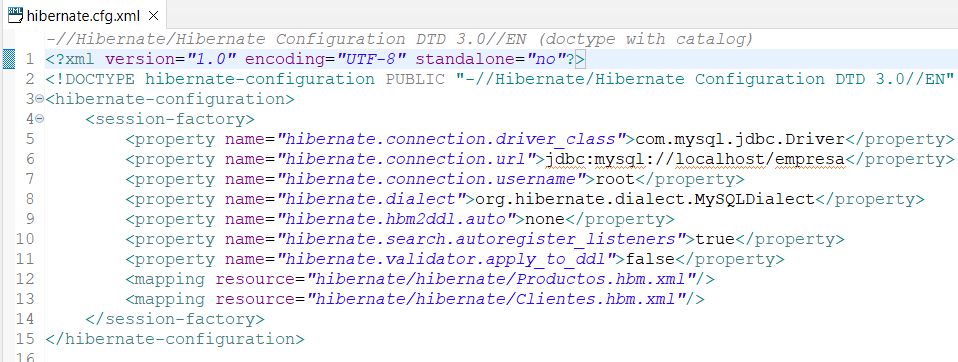
**Podremos observar que se nos han creado los siguientes ficheros con la información de las tablas de la base de datos.**

****

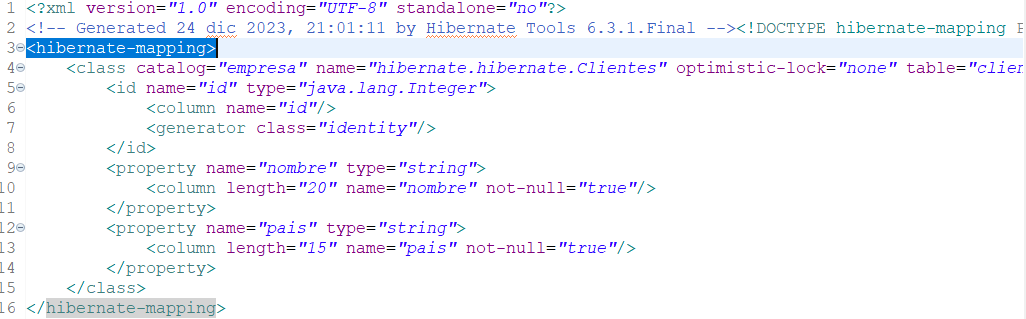
**Esta es la clase de Clientes que contiene los campos de la tabla clientes.**

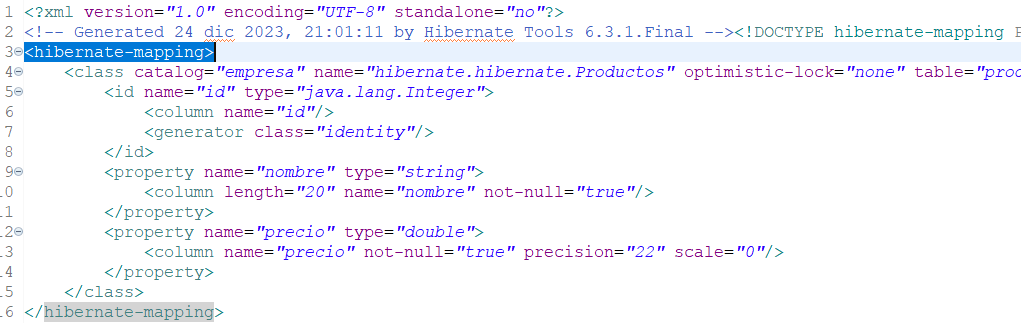
****

**La clase Productos que contendrá los campos de la tabla productos.**

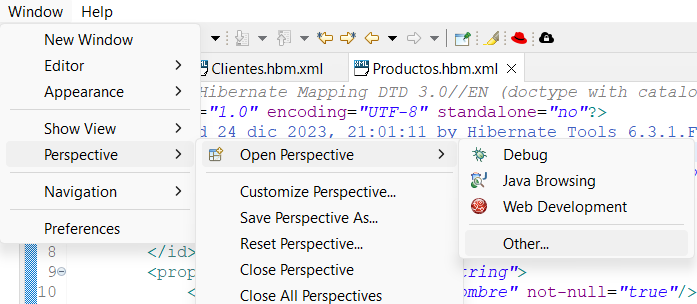
****

**En el archivo de condiguracion.xml podremos ver que se han añadido nuevas líneas sobre el mapeo de los archivos.**

****

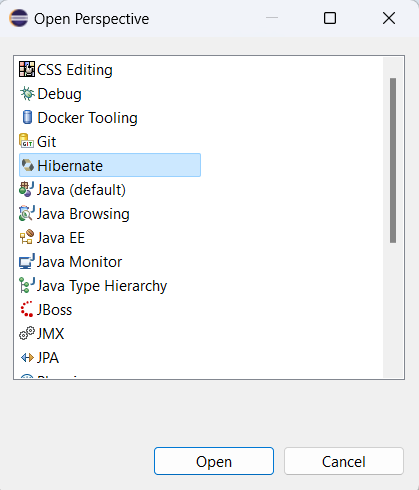
****

**Los archivos de Clientes y Productos .hbm.xml con información sobre el mapeo de las clases con las tablas.**

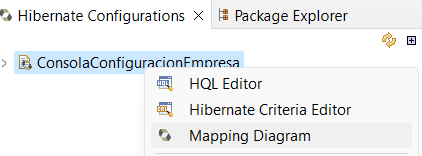
****

**Abrimos la perspectiva de Hibernate para comprobar que la conexión con la base de datos funciona correctamente.**

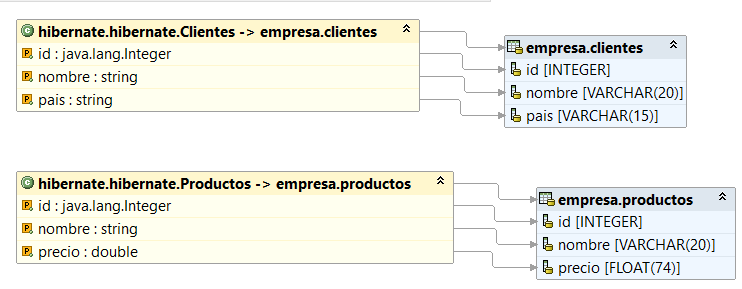
**Window > Perspective > Open Perspective > Other…**

****

**Se nos abrirá la siguiente ventana en la cual seleccionamos Hibernate.**

****

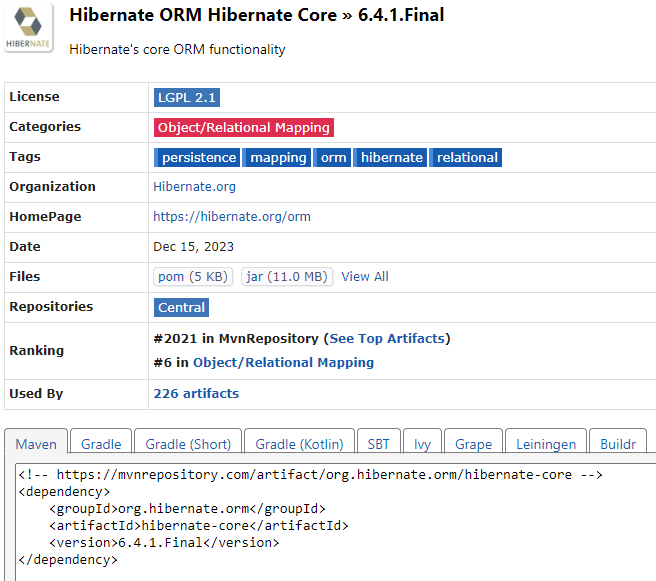
**En la parte izquierda podremos observar que apareció otra pestaña llamada Hibernate Configurations le damos clic derecho sobre la consola que hemos creado en los pasos anteriores y después a la opción que pone Mapping Diagram para que nos cree el diagrama.**

****

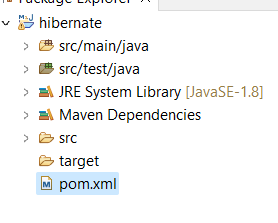
**Este seria el diagrama que se ha creado sobre nuestras tablas de la base de datos que como podremos observar ha funcionado correctamente.**

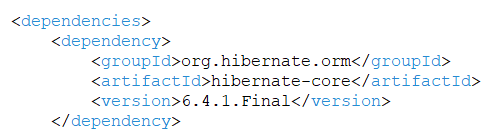
## **Buscamos las dependencias de Hibernate ORM Hibernate Core última versión.**

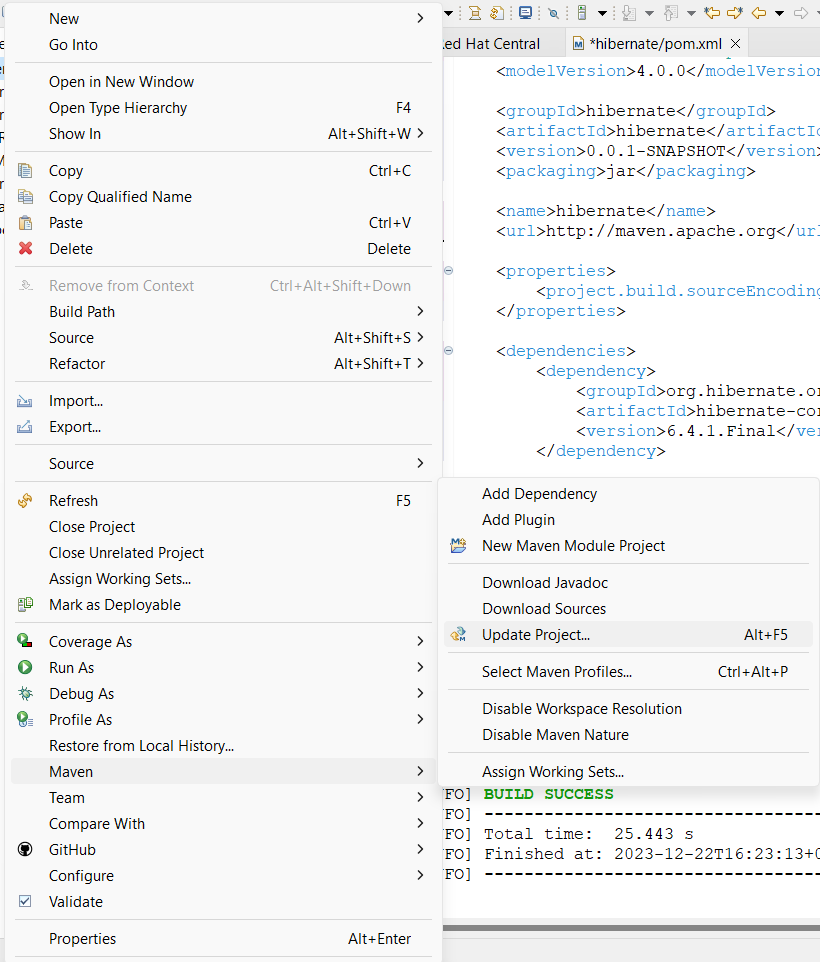
[**https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-core**](https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-core)

****

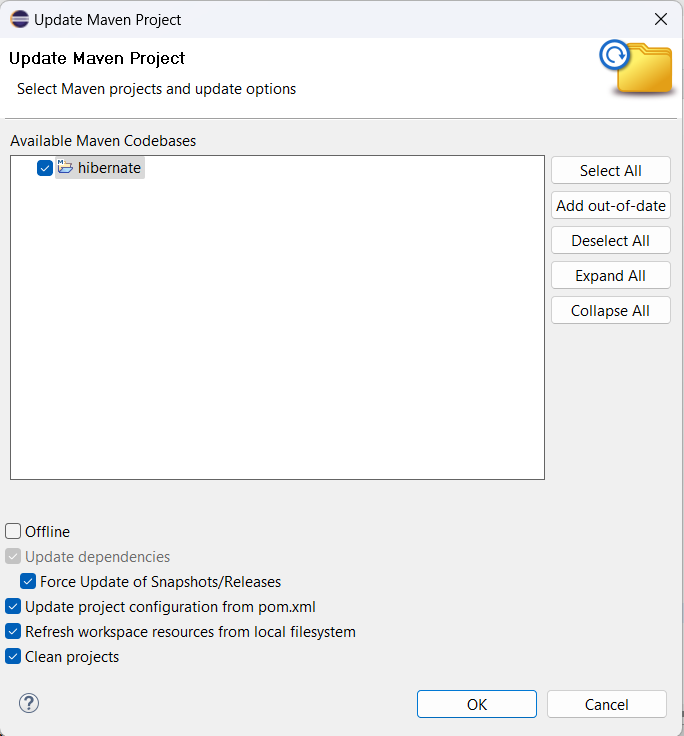
**Copiamos las dependencias en el pom de nuestro proyecto Maven.**

****

****

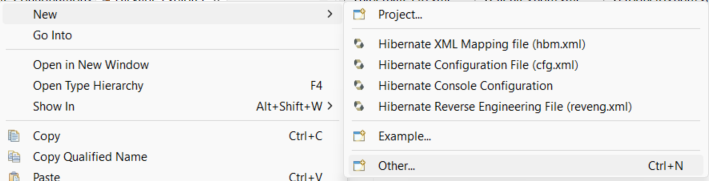
****

**Clic derecho sobre el proyecto > Maven > Update Project.**

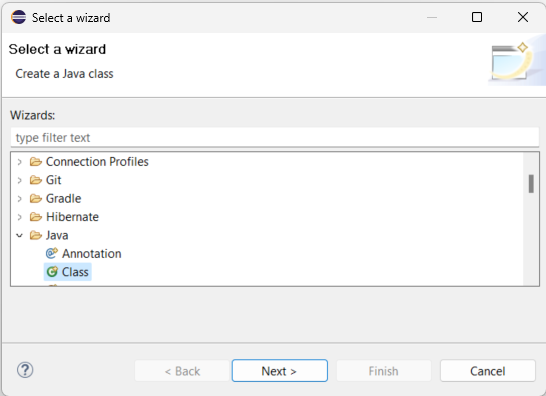
****

**Le damos al Check Force Update of Snapshots/Releases y luego a OK. Para que se actualicen las dependencias y bajen la nueva dependencia.**

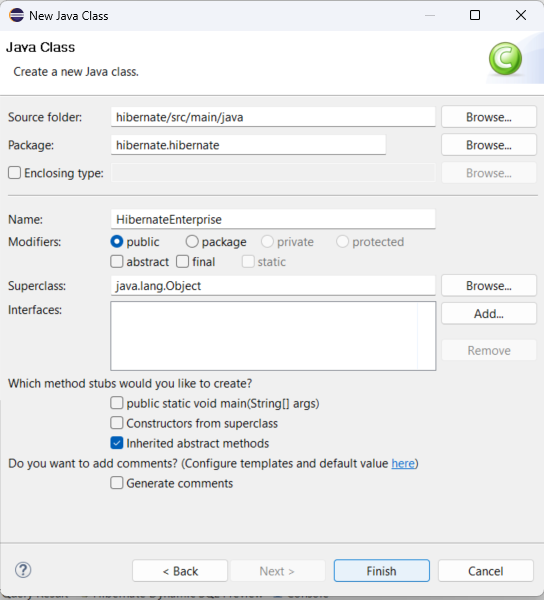
## **Usar Hibernate para hacer operaciones en la Base de Datos.**

****

**Le damos clic derecho sobre el proyecto luego New > Other…**

****

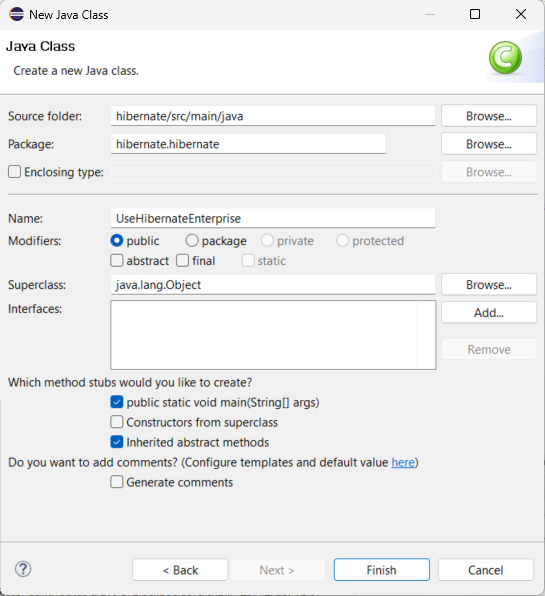
**Luego nos aparecerá esta ventana para seleccionar vamos a Java > Class > Next**

****

**Le ponemos el nombre de HibernateEnterprise y lo creamos.**

|  |
| --- |
| **Package hibernate.hibernate;**  **import java.util.Iterator;**  **import java.util.List;**  **import org.hibernate.HibernateException;**  **import org.hibernate.ObjectNotFoundException;**  **import org.hibernate.Session;**  **import org.hibernate.SessionFactory;**  **import org.hibernate.Transaction;**  **import org.hibernate.cfg.Configuration;**  **public class HibernateEnterprise {**  **private static SessionFactory *sf*; // this SessionFactory esi be created once and used for all the connections**  **private static Productos *p*;**  **HibernateEnterprise() {// constructor**  **// sf = HibernateUtil.getSessionFactory();**  ***sf* = new Configuration().configure().buildSessionFactory(); // also esió**  **}**  **public void close() {**  ***sf*.close();**  **}**  **public void addProduct(String name, double esió) {**  **Session esión = *sf*.openSession();// esión es la variable que tiene el método**  **// esi para guardar productos**  **Transaction tx = null;**  **// esión the esióno with the parameters in the method**  **Productos p = new Productos();**  **p.setNombre(name);**  **p.setPrecio(esió);**  **// keep it in the database=⎝session.save(p)**  **try {**  **System.*out*.println(“======================================”);**  **System.*out*.printf(“Insertando la Fila en la Base de Datos: %s, %s\n”, name, esió);**  **System.*out*.println(“======================================”);**  **tx = esión.beginTransaction();**  **esión.~~save~~(p);// we INSERT p into the table PRODUCTS**  **tx.commit();// if esión.save doesnt produce an exception, we commit; the transaction**  **} catch (Exception e) {// if there is any exception, we “rollback” and close safely**  **if (tx ¡= null) {**  **tx.rollback();**  **}**  **} finally {**  **esión.close();**  **}**  **}**  **public void showProducts() {**  **Session esión = *sf*.openSession();**  **Transaction tx = null;**  **try {**  **tx = esión.beginTransaction();**  **List allproducts = esión.~~createQuery~~(“From Productos”).list();**  **Iterator it = allproducts.iterator();**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Buscando Productos…”);**  **System.*out*.println(“======================================”);**  **while (it.hasNext()) {**  **// for (Iterator = allproducts.iterator(); iterator.hasNext()😉{**  **Productos p = (Productos) it.next();**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Id: “ + p.getId());**  **System.*out*.println(“Nombre: “ + p.getNombre());**  **System.*out*.println(“Precio: “ + p.getPrecio());**  **System.*out*.println(“======================================”);**  **}**  **tx.commit();**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Finalizada la Busqueda…”);**  **System.*out*.println(“======================================”);**  **} catch (HibernateException e) {**  **if (tx ¡= null)**  **tx.rollback();**  **e.printStackTrace();**  **} finally {**  **esión.close();**  **}**  **}**  **public Productos findProductById(int id) {**  **Session esión = *sf*.openSession();**  **Transaction tx = null;**  **Productos p = new Productos();**  **try {**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Cargando Producto de la Base de Datos…”);**  **System.*out*.println(“======================================”);**  **tx = esión.beginTransaction();**  **p = (Productos) esión.~~load~~(Productos.class, id);**  **tx.commit();**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Producto con ID -> “ + id);**  **System.*out*.println(“Su Nombre es -> “ + p.getNombre());**  **System.*out*.println(“======================================”);**  **} catch (ObjectNotFoundException e) {**  **if (tx ¡= null) {**  **System.*out*.println(e);**  **System.*out*.println(“Product not found”);**  **}**  **} catch (Exception e) {**  **if (tx ¡= null) {**  **System.*out*.println(e);**  **tx.rollback();**  **}**  **} finally {**  **esión.close();**  **}**  **return p;**  **}**  **public void deleteProductById(int id) {**  **Productos p = new Productos();**  **Session esión = *sf*.openSession();**  **Transaction tx = null;**  **try {**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Buscando Producto con ID -> “ + id);**  **System.*out*.println(“======================================”);**  **tx = esión.beginTransaction();**  **p = (Productos) esión.get(Productos.class, id);**  **if (p ¡= null) {**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Borrando Producto de la Base de Datos…”);**  **System.*out*.println(“======================================”);**  **esión.~~delete~~(p);**  **tx.commit();**  **System.*out*.println(“======================================”);**  **System.*out*.printf(**  **“Producto Borrado de la Base de Datos …” + “\n ID -> %s\n Nombre -> %s\n Precio -> %s”,**  **p.getId(), p.getNombre(), p.getPrecio());**  **System.*out*.println(“\n======================================”);**  **} else {**  **System.*out*.println(“======================================”);**  **System.*out*.println(“No Se Encontro Ningun Producto con ID -> “ + id);**  **System.*out*.println(“======================================”);**  **}**  **} catch (Exception e) {**  **if (tx ¡= null) {**  **tx.rollback();**  **}**  **} finally {**  **esión.close();**  **}**  **}**  **public void updateProductById(int id, String newName, double newPrice) {**  **Productos p = new Productos();**  **Session esión = *sf*.openSession();**  **Transaction tx = null;**  **try {**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Modificando el Producto de la Base de Datos…”);**  **System.*out*.println(“Con los Siguientes Datos…”);**  **System.*out*.println(“ID -> “ + id);**  **System.*out*.println(“Nombre -> “ + newName);**  **System.*out*.println(“Precio -> “ + newPrice);**  **System.*out*.println(“======================================”);**  **tx = esión.beginTransaction();**  **p = (Productos) esión.~~load~~(Productos.class, id);// we load the esióno**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Datos del Producto en la Base de Datos…”);**  **System.*out*.println(“======================================”);**  **System.*out*.printf(“ ID -> %s\n Nombre -> %s\n Precio -> %s”, p.getId(), p.getNombre(), p.getPrecio());**  **System.*out*.println(“\n======================================”);**  **p.setPrecio(newPrice);// we change the properties**  **p.setNombre(newName);**  **esión.~~update~~(p);// we update the values in the database**  **tx.commit();**  **System.*out*.println(“======================================”);**  **System.*out*.println(“Producto Modificado”);**  **System.*out*.println(“======================================”);**  **System.*out*.printf(“Datos del Producto Modificado…” + “\n ID -> %s\n Nombre -> %s\n Precio -> %s”,**  **p.getId(), p.getNombre(), p.getPrecio());**  **System.*out*.println(“\n======================================”);**  **} catch (Exception e) {**  **System.*out*.println(“======================================”);**  **System.*out*.println(“No Se Encontro el Producto con ID -> “ + id);**  **System.*out*.println(“======================================”);**  **if (tx ¡= null) {**  **tx.rollback();**  **}**  **} finally {**  **esión.close();**  **}**  **}**  **}** |

**En esta clase tenemos la lógica de las operaciones que haremos con la base de datos. Como por ejemplo para crear una sesión , cerrar la sesión, añadir un producto a la base de datos, mostrar todos los productos de la base de datos, buscar un producto introduciendo el ID del producto, borrar un producto por ID y por último modificar un producto por id introduciendo su nuevo nombre y precio.**

****

**Creamos una nueva clase para probar los métodos de HibernateEnterprise y para poder probarlos creamos la nueva clase como main y seleccionamos la opción public static void main(String[] args) para que se nos cree automáticamente.**

|  |
| --- |
| **package hibernate.hibernate;**  **import java.util.logging.Level;**  **import java.util.logging.LogManager;**  **public class UseHibernateEnterprise {**  **public static void main(String[] args) {**  **LogManager.*getLogManager*().getLogger("").setLevel(Level.*SEVERE*);**  **HibernateEnterprise h = new HibernateEnterprise();**  **// System.out.println("");**  **// h.addProduct("monitor",170);**  **// System.out.println("");**  **// h.showProducts();**  **// System.out.println("");**  **// h.findProductById(3);**  **// System.out.println("");**  **// h.deleteProductById(7);**  **// h.showProducts();**  **System.*out*.println("");**  **h.updateProductById(5,"ssd",105);**  **h.updateProductById(8,"ssd",165);**  **h.close();**  **}**  **}** |

**Dentro de esta clase creamos un LogManager para que no nos aparezcan los logs del servidor en la consola , después creamos un objeto de la clase y vamos probando cada uno de los métodos para comprobar que todo funciona correctamente y por último cerramos la conexión para que no se pierda ningún dato.**