HIBERNATE

eclipse

HIBERNATE EN ECLIPSE ENTERPRISE

David Cristian Pirosca

Índice

Proyecto Hibernate en Eclipse 1

Base de Datos empresa. 1

Instalamos Plugin JBoss Tools del Matketplace. 1

Creamos un nuevo Proyecto Maven. 3

Descargamos el Jar del Conector y Creamos una Conectividad con la Base de Datos. 5

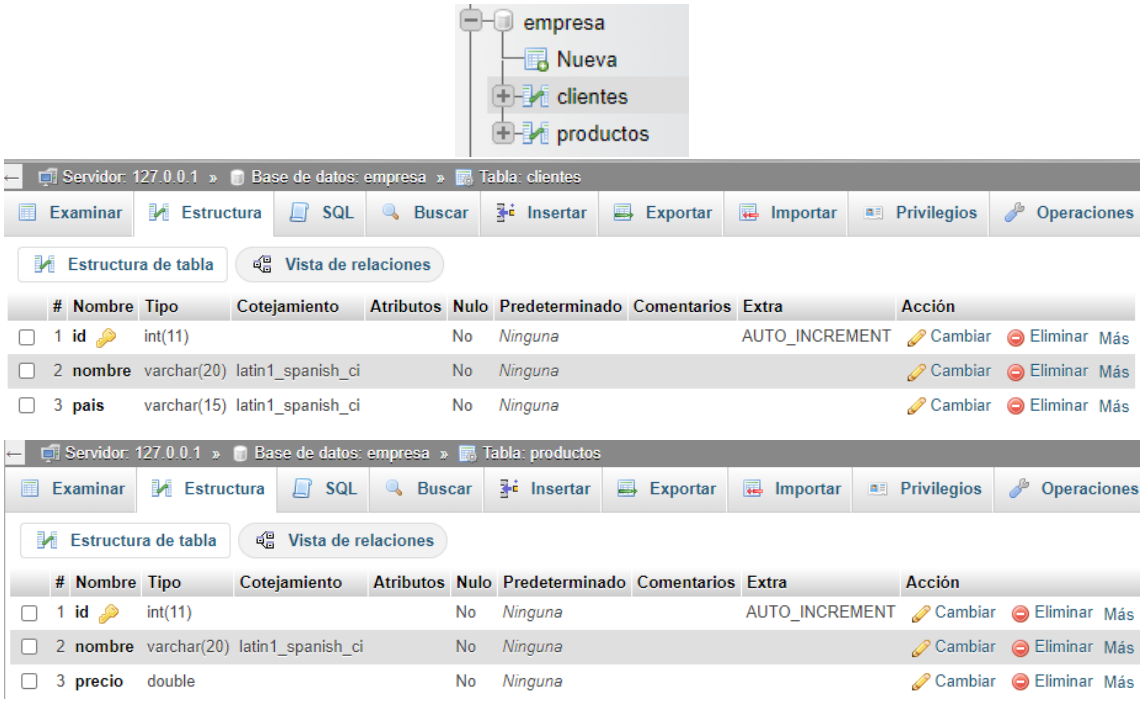
Creamos los archivos de Hibernate para poder hacer un mapeo de objetos 11

Buscamos las dependencias de Hibernate ORM Hibernate Core última versión. 23

Usar Hibernate para hacer operaciones en la Base de Datos. 25

Proyecto Hibernate en Eclipse

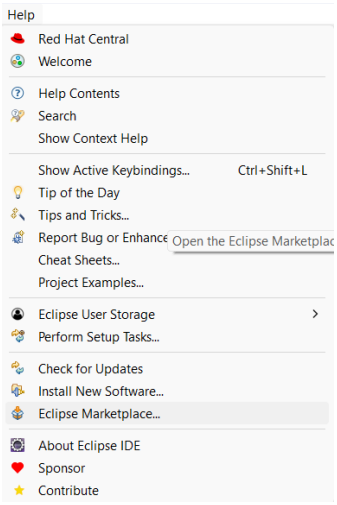
Base de Datos empresa.



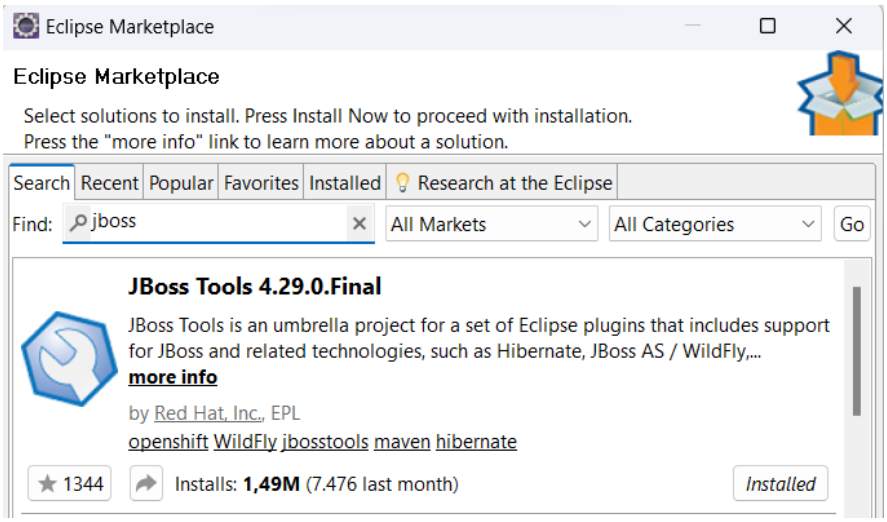
Descargar empresa.sql

https://github.com/davidpirosca/davidpirosca.github.io/blob/main/Codigos/Java/archivos/empresa.sql

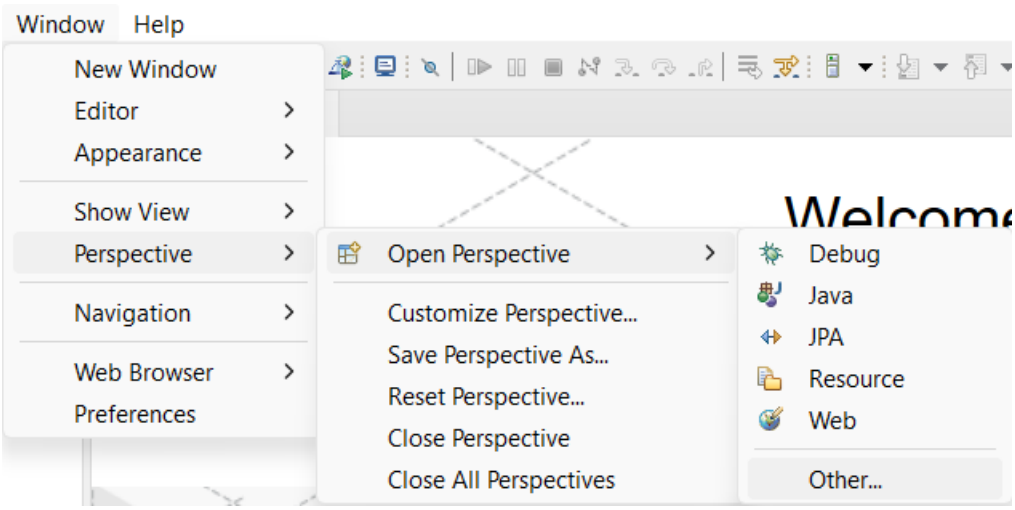
Instalamos Plugin JBoss Tools del Matketplace.



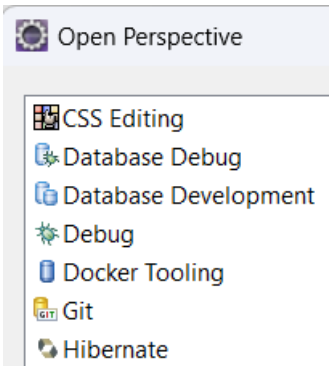
- **Help > Eclipse Marketplace**



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

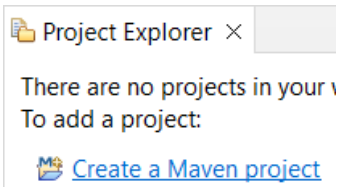


- **Comprobamos que está instalado Hibernate en…**
  Window > Perspective > Open Perspective > Other…

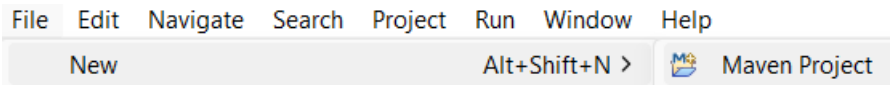


- **Podremos observar que está instalado.**

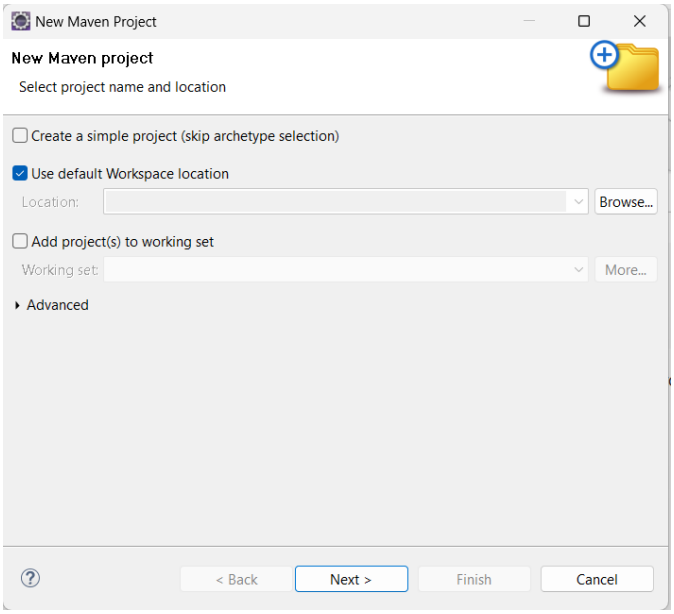
Creamos un nuevo Proyecto Maven.



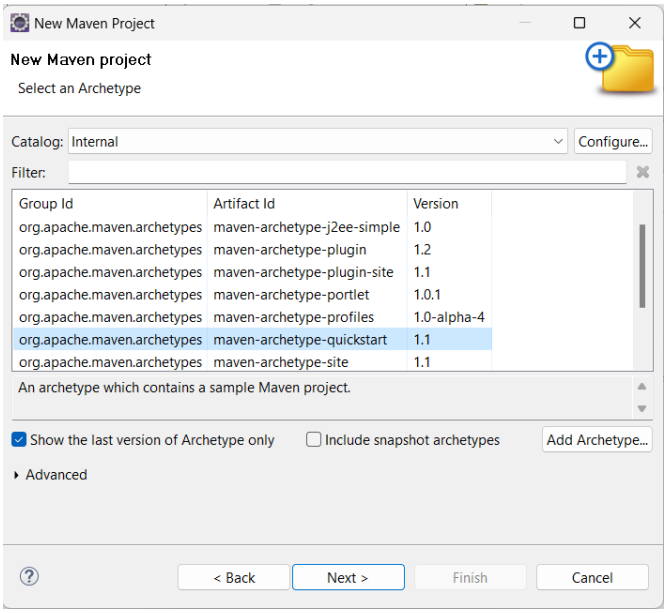
- **En la parte izquierda nos aparece para crear un nuevo proyecto si es nuevo el espacio de trabajo.**



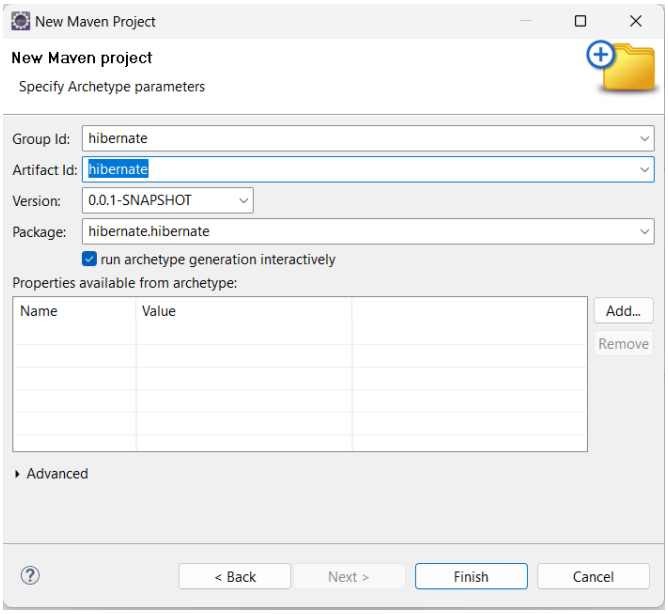
- **En el caso de que el espacio de trabajo no sea nuevo tendremos que crearlo desde File > New > Maven Project**
  ***(Si no está lo buscamos en Other… > Maven > Maven Project).**



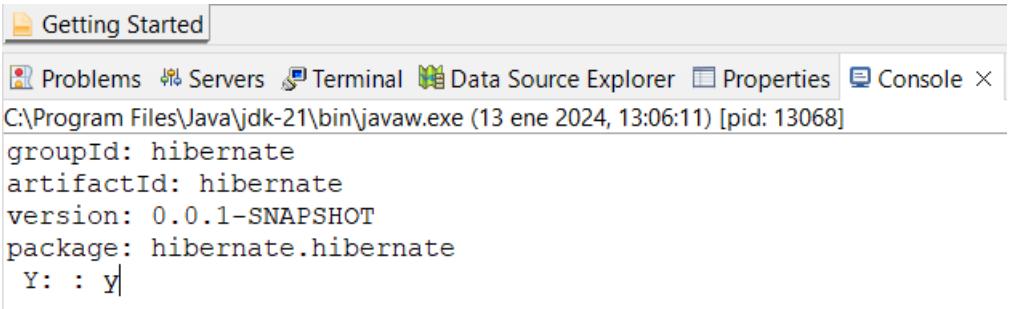
- **En este caso lo guardamos en la ubicación del Espacio de Trabajo si quisiéramos otra ubicación lo podríamos cambiar y le damos a next.**



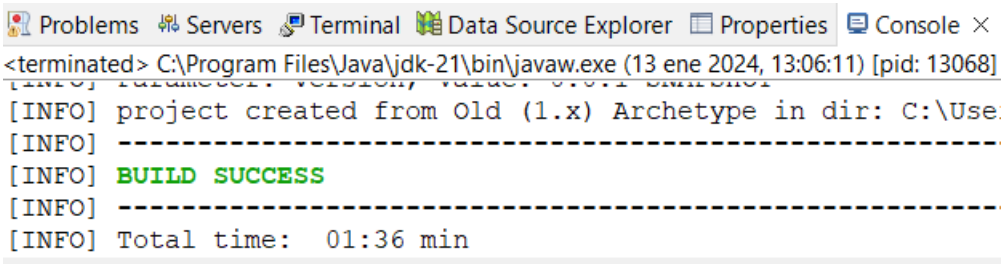
- **En Catalog seleccionamos Internal y después buscamos la opción “maven-archetype-quickstart”.**



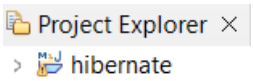
- **Le ponemos id y le damos a finish.**



- **En la parte de abajo del ide esta la consola que nos piden confirmación para la creación introducimos "y" y le damos a la tecla Intro para que se cree el proyecto con Maven.**



- **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.

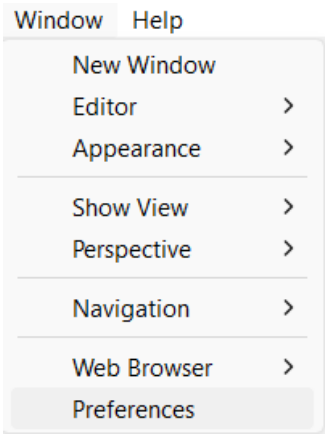
https://mvnrepository.com/artifact/com.mysql/mysql-connector-j

MySQL Connector/J » 8.2.0

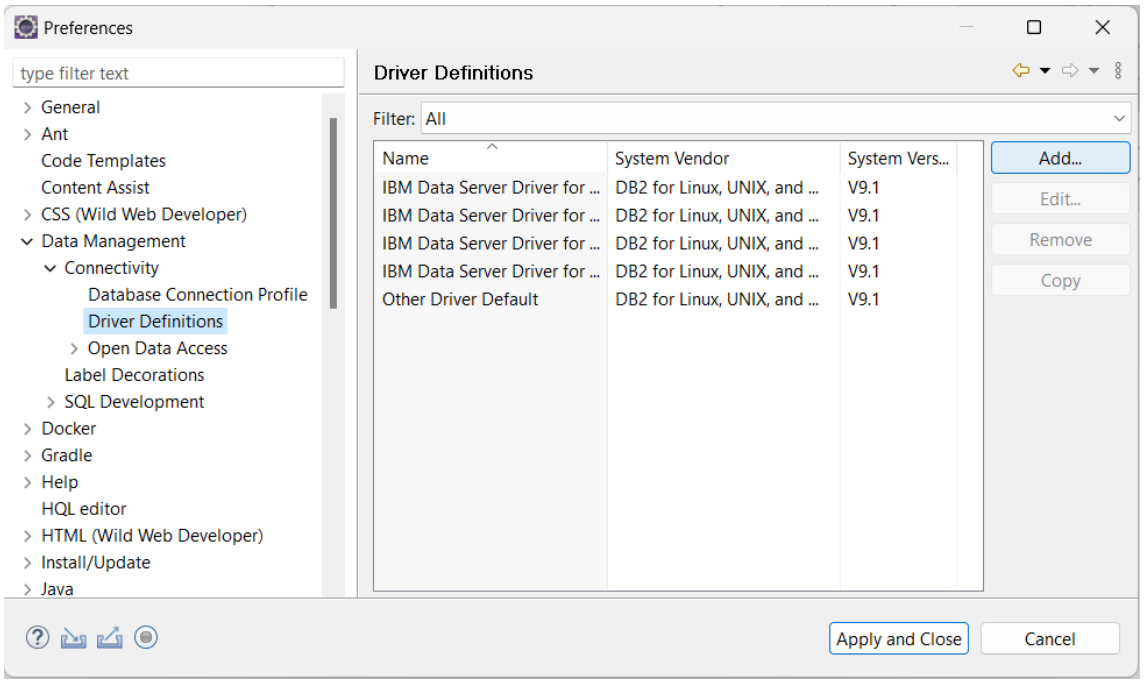
MySQL Connector/J is a JDBC Type 4 driver, which means that it is pure Java implementation of the MySQL protocol and does not rely on the MySQL client libraries. This driver supports auto-registration with the Driver Manager, standardized validity checks, categorized SQLExceptions, support for large update counts, support for local and offset date-time variants from the java.time package, support for JDBC-4.x XML processing, support for per connection client information and support for the NCHAR, NVARCHAR ...

Categories	JDBC Drivers
Tags	database sql jdbc driver connector rdbms mysql connection
Organization	Oracle Corporation
HomePage	http://dev.mysql.com/doc/connector-j/en/
Date	Oct 25, 2023
Files	pom (3 KB) jar (2.4 MB) View All
Repositories	Central
Ranking	#1136 in MvnRepository (See Top Artifacts) #10 in JDBC Drivers
Used By	420 artifacts

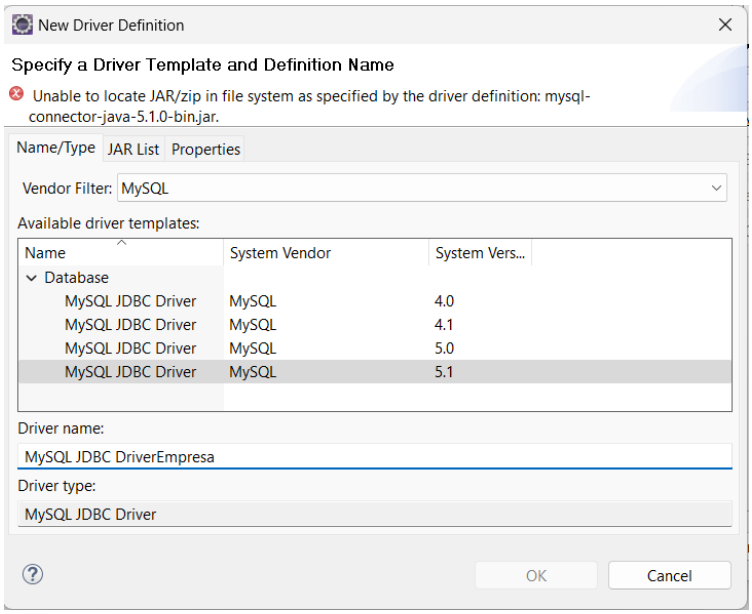
- **Nos descargamos el Jar de la última versión. Lo podremos guardar donde queramos, pero sería aconsejable que este en la carpeta del proyecto para que cuando se exporte el proyecto se exporten también los archivos jar del mismo.**



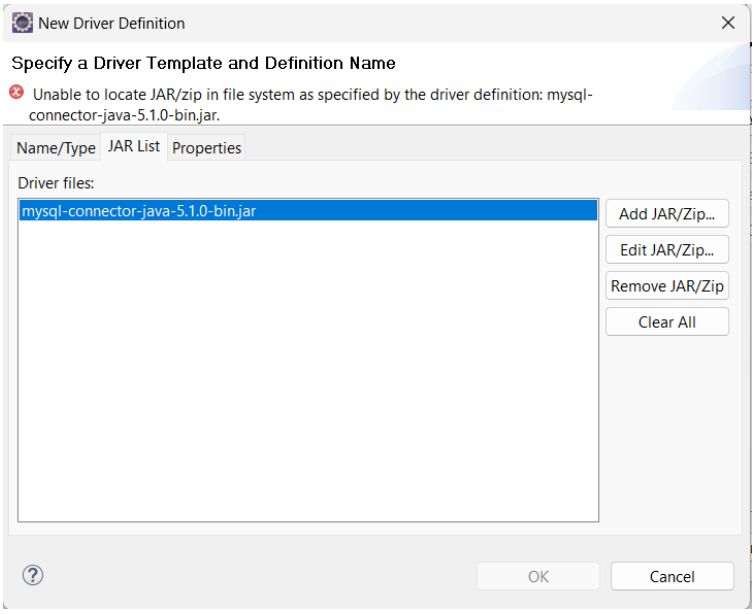
- **Window > Preference**



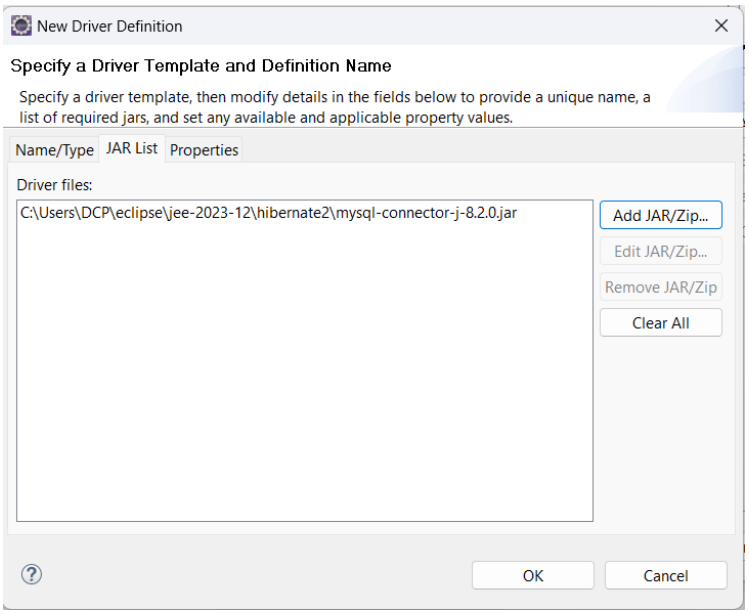
- **Data Management > Conectivity > Driver Definitions.**
  Para que podamos agregar una nueva conectividad a la base de datos. Le damos a Add.



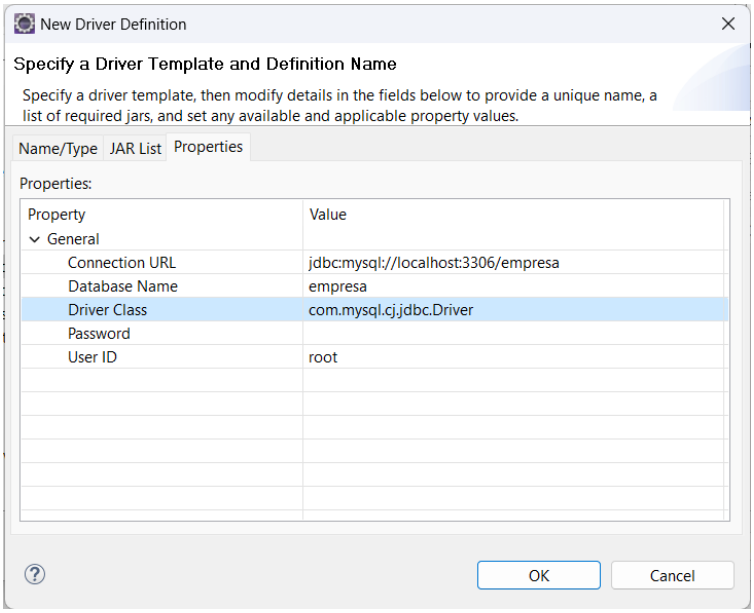
- **En Vendor Filter seleccionamos MySQL y en los Drivers seleccionamos el ultimo que en este caso seria el 5.1 y seria conveniente modificar el nombre para saber cual es en este caso le agregamos Empresa al final para distinguirlo.**



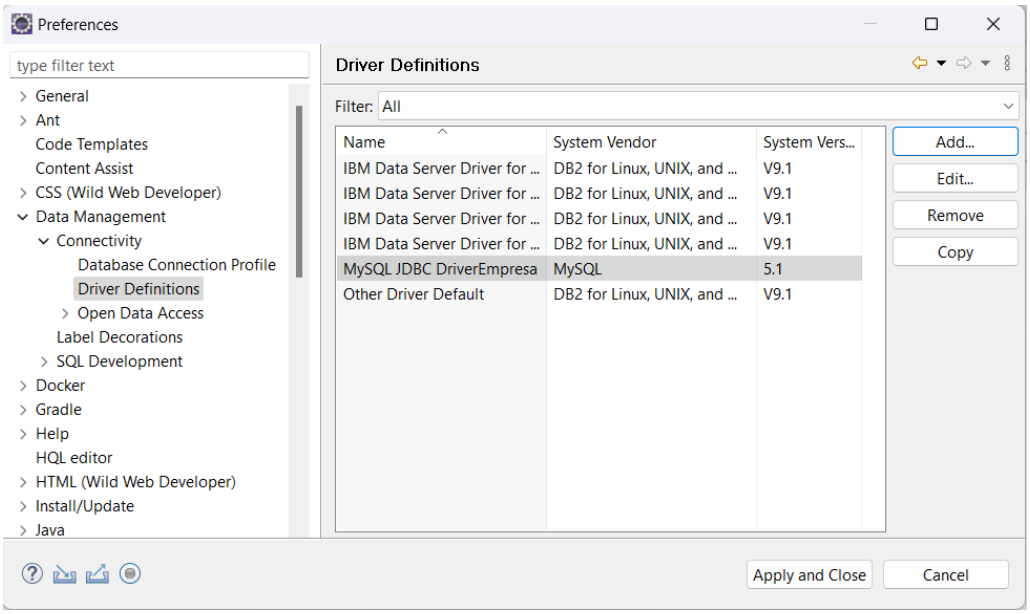
- **Nos desplazamos a la siguiente pestaña que se llama JAR List y nos aparecerá un driver que es el que hemos seleccionado nosotros antes y dice que no lo encuentra, pero a nosotros eso nos da igual ya que nosotros usaremos el que hemos descargado antes, seleccionamos el driver y le damos a Remove JAR/ZIP.**



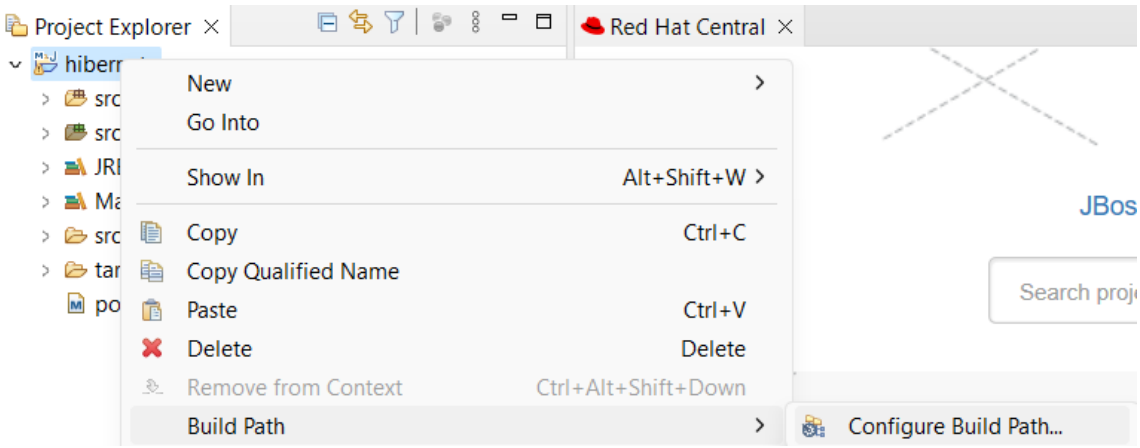
- **Le damos a Add JAR/ZIP y buscamos el conector que hemos descargado en el paso anterior y le damos a abrir. Y como podremos observar ha desaparecido el Warning que aparecía en la parte superior porque no encuentra del conector.**



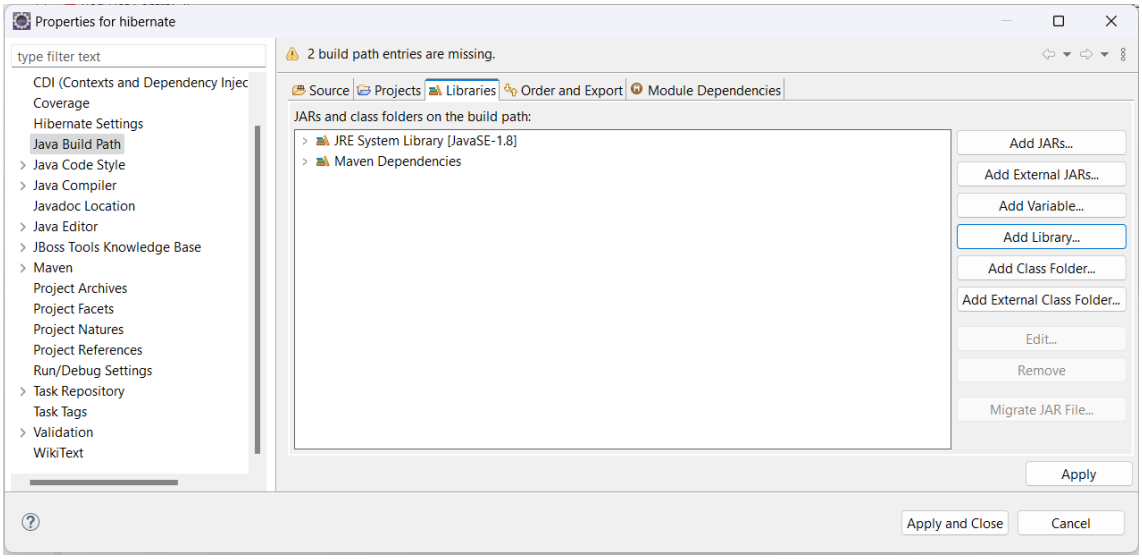
- **Nos desplazamos a la siguiente pestaña que seria Properties y le metemos los datos de conexión a la base de datos y en el Driver Class ponemos el del conector nuevo que en este caso es com.mysql.cj.jdbc.Driver. Y por último le damos a Ok.**



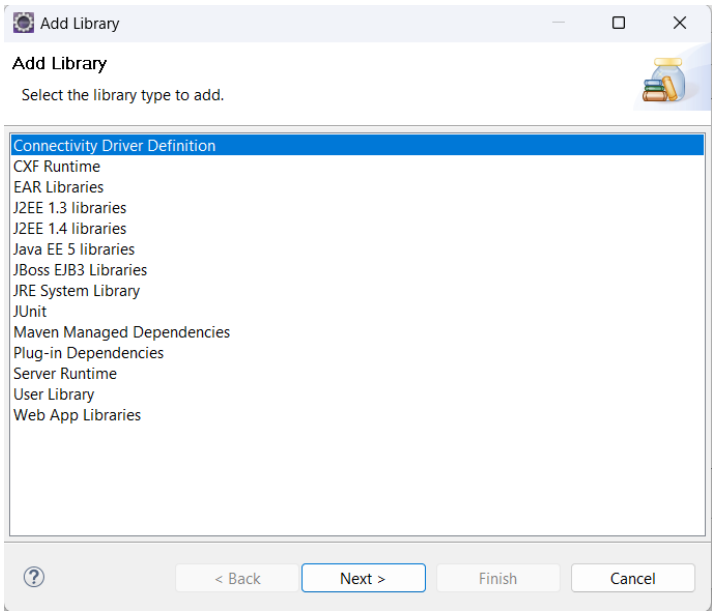
Como podremos observar se ha creado correctamente le damos a Apply and Close.



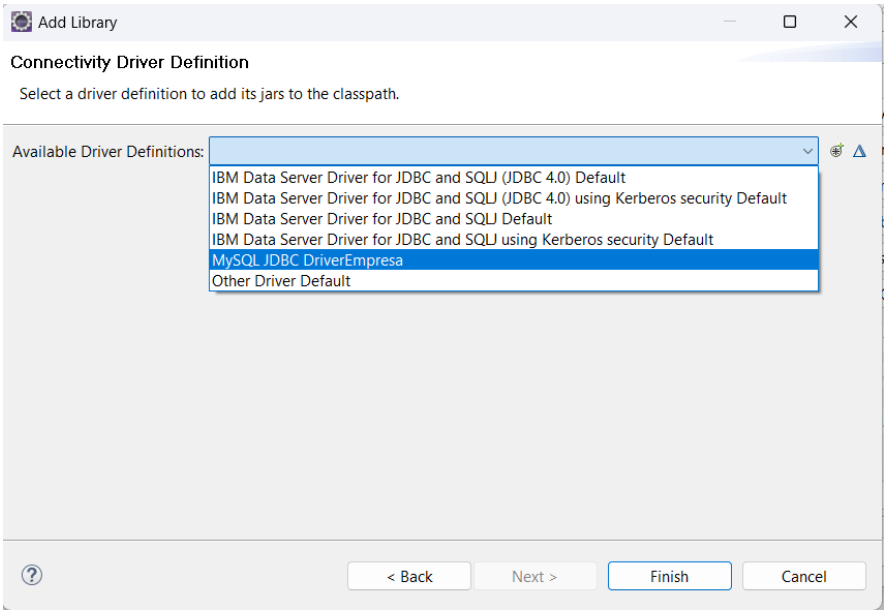
Hacemos clic derecho sobre el proyecto > Build Path > Configure Build Path…



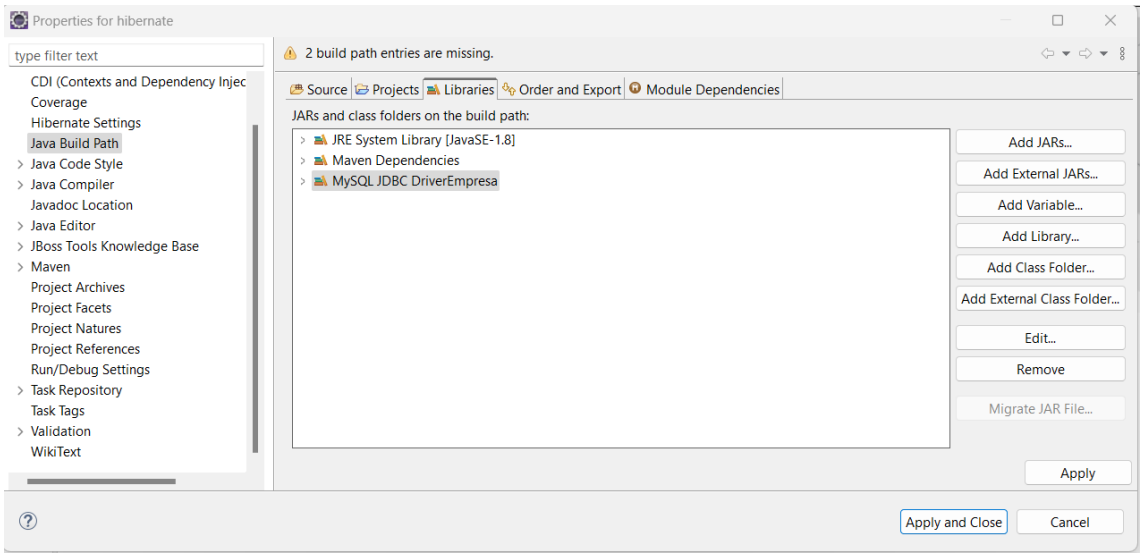
- **Nos desplazamos a la pestaña de Libraries y le damos a Add Library...**



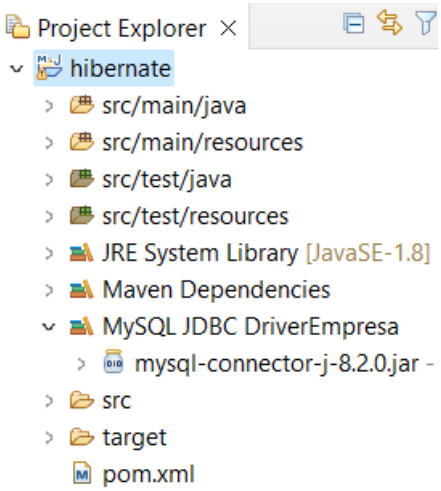
- **Seleccionamos la opción Connectivity Driver Definition y le damos a Next.**



- **En la siguiente ventana le damos al desplegable y elegimos lo que acabamos de crear que nos aparecerá entre las opciones, por último, le damos a Finish.**

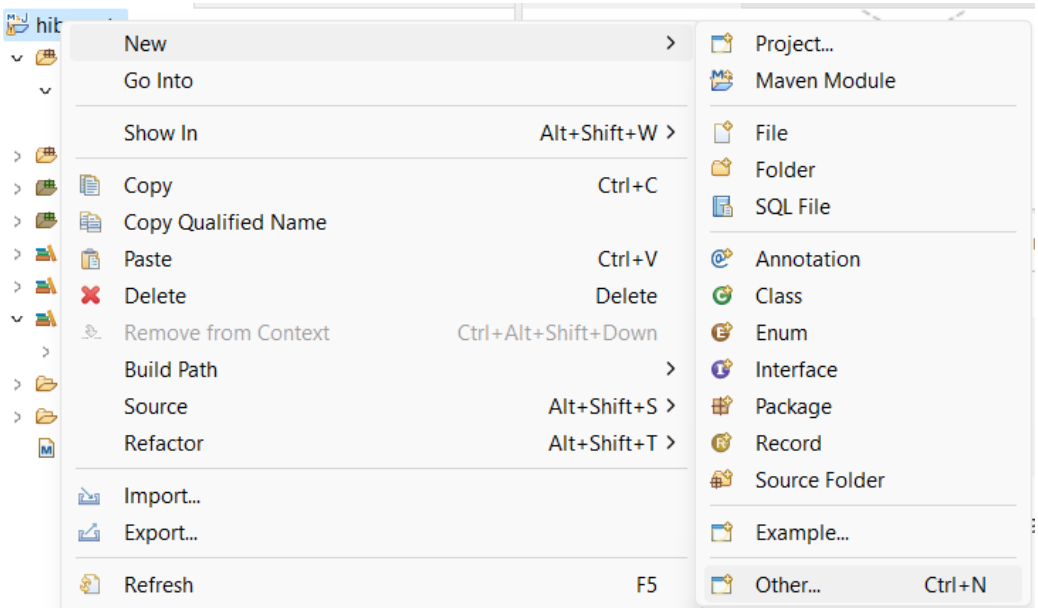


- **Le damos a Apply y después a Apply and Close.**

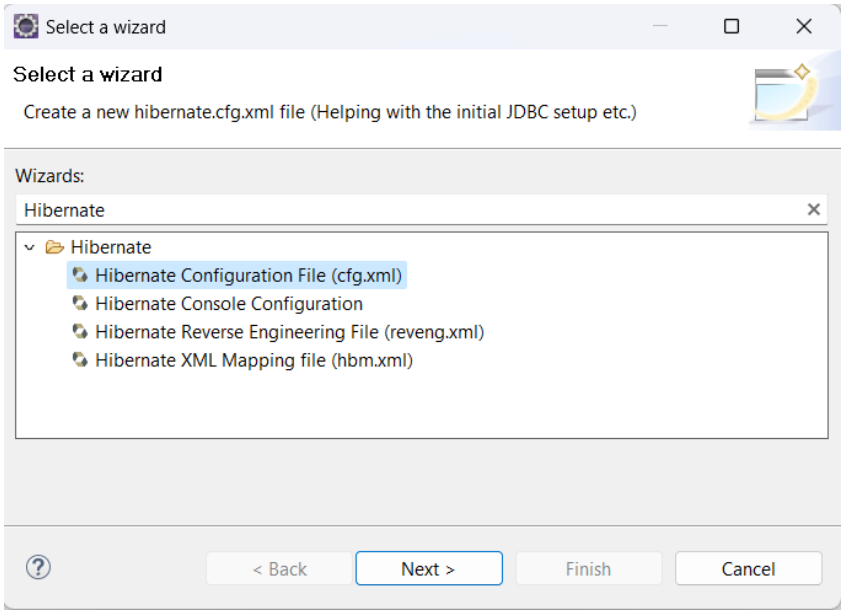


- **En la parte izquierda donde está el explorador podremos ver que tenemos la nueva librería con nuestro conector jar.**

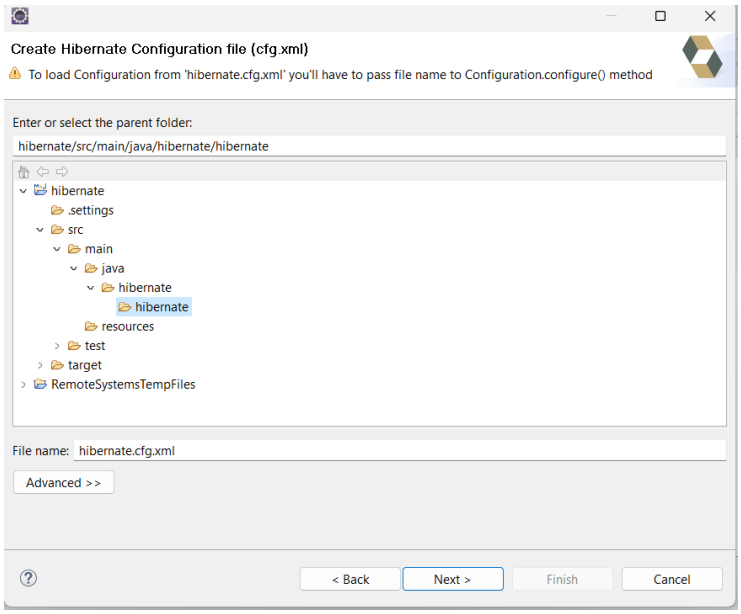
Creamos los archivos de Hibernate para poder hacer un mapeo de objetos



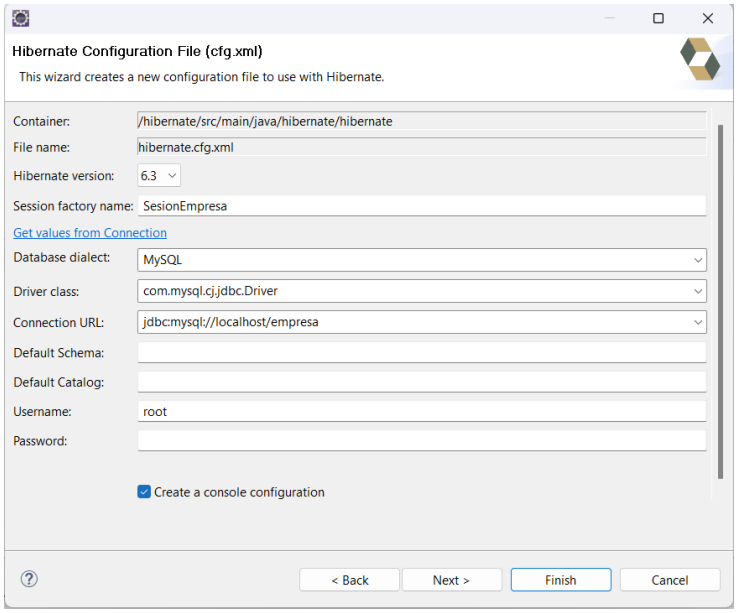
Ahora vamos a crear y configurar el archivo de Configuración de Hibernate. Clic Derecho sobre Proyecto > New > Other…



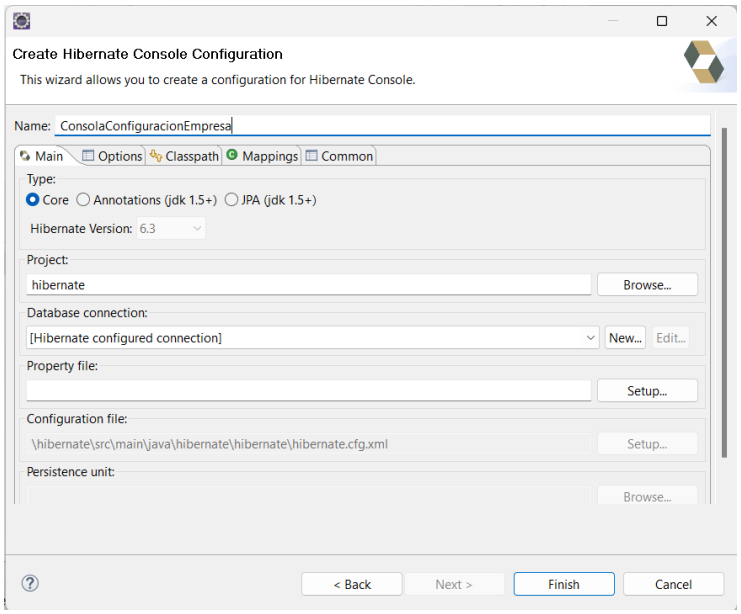
Ponemos en el buscador Hibernate o buscamos la carpeta de Hibernate y dentro tendremos Hibernate Configuration File (cfg.xml) y le damos a Next.



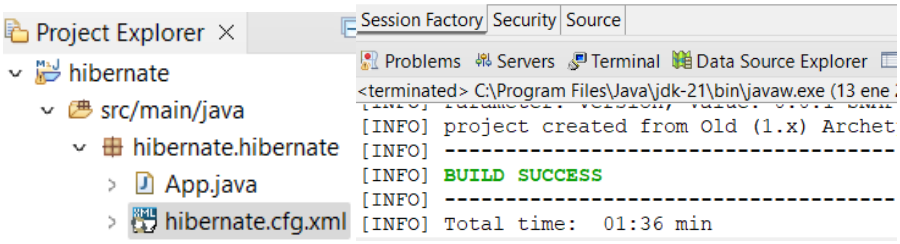
Seleccionamos la carpeta donde queremos que se guarde el archivo y un nombre identificativo para el archivo que este caso para no tener problemas lo guardamos en el paquete del proyecto que se nos ha creado por defecto y le damos a Next.



Le ponemos un nombre que en este caso lo llamaremos SesionEmpresa seleccionamos el lenguaje que seria MySQL el driver el nuestro es nuevo y no sale, pero funciona porque nuestro conector lo tiene y lo reconoce que sería el siguiente com.mysql.cj.jdbc.Driver y la url que sería jdbc:mysql://localhost/empresa le ponemos usuario y contraseña. Por último, seleccionamos la opción Create a console configuration y le damos a Next.



- **Le ponemos un nombre a la consola de configuración que en este caso sería ConsolaConfiguracionEmpresa y le damos a Finish.**

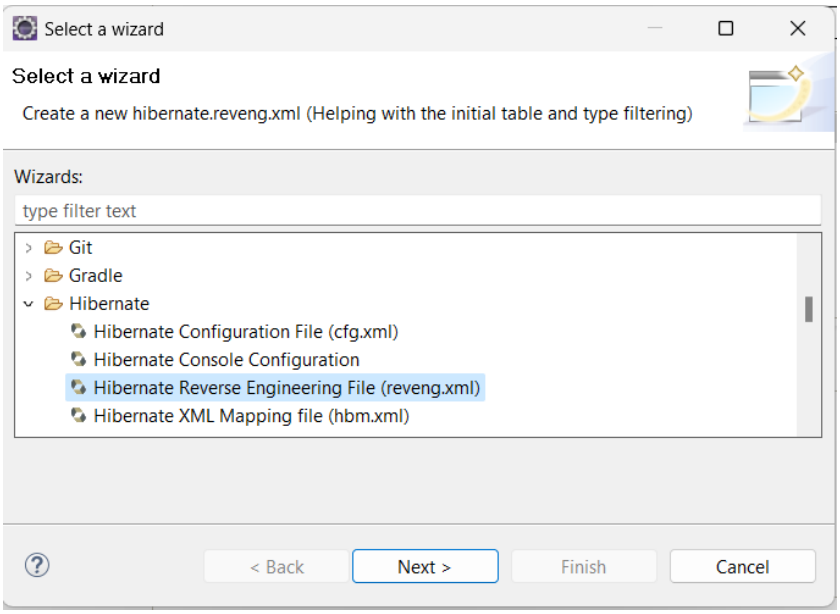


- **En la parte izquierda podremos observar que se nos ha creado el archivo de configuración si le damos doble clic se nos abrirá, pero tenemos que ir a la parte de abajo que es donde están las vistas del archivo ya que tiene varias y seleccionamos la de Source para que podamos ver el código del archivo que le hemos puesto al crearlo y si necesitásemos cambiar algún dato del archivo se podría hacer editando el archivo.**

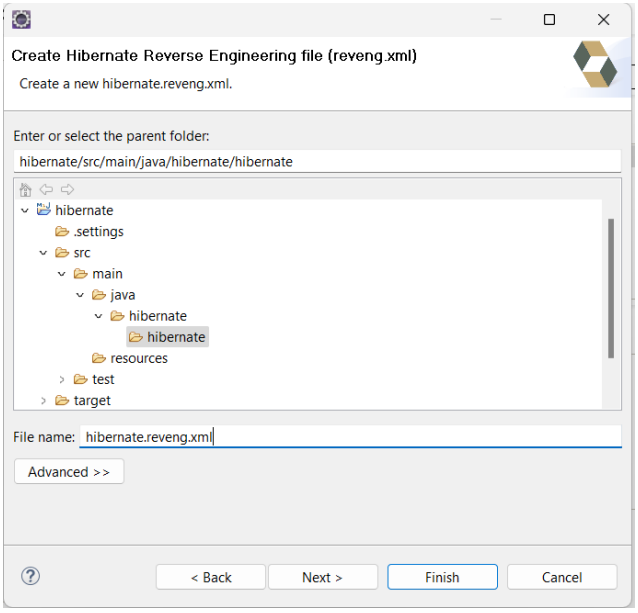
hibernate.cfg.xml

```
-//Hibernate/Hibernate Configuration DTD 3.0//EN (doctype with catalog)
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory name="SesionEmpresa">
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost/empresa</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    </session-factory>
</hibernate-configuration>
```

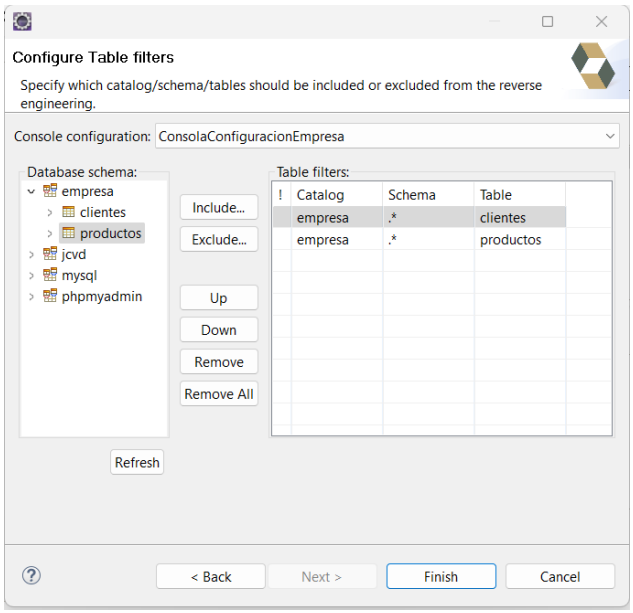
- **Los datos del archivo se pueden ver en el código.**



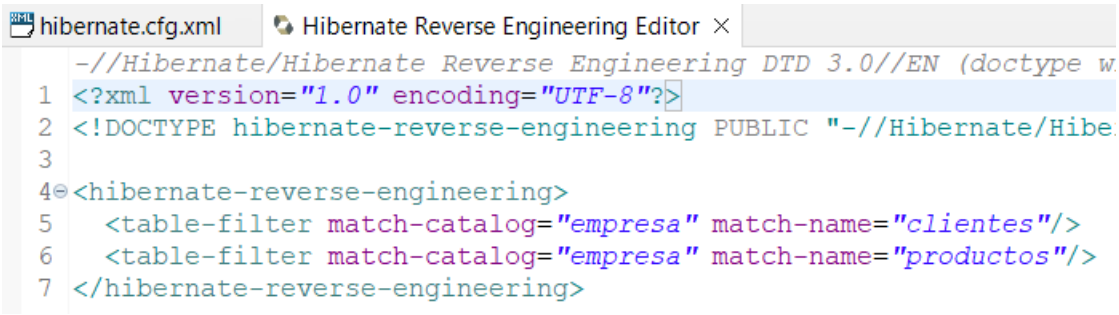
- **Vamos a crear un nuevo archivo para la ingeniería inversa que lo que hará es crearnos clases con las tablas de la base de datos.**
  Clic Derecho sobre Proyecto > New > Other… > Hibernate > Hibernate Reverse Engineering File (reveng.xml).



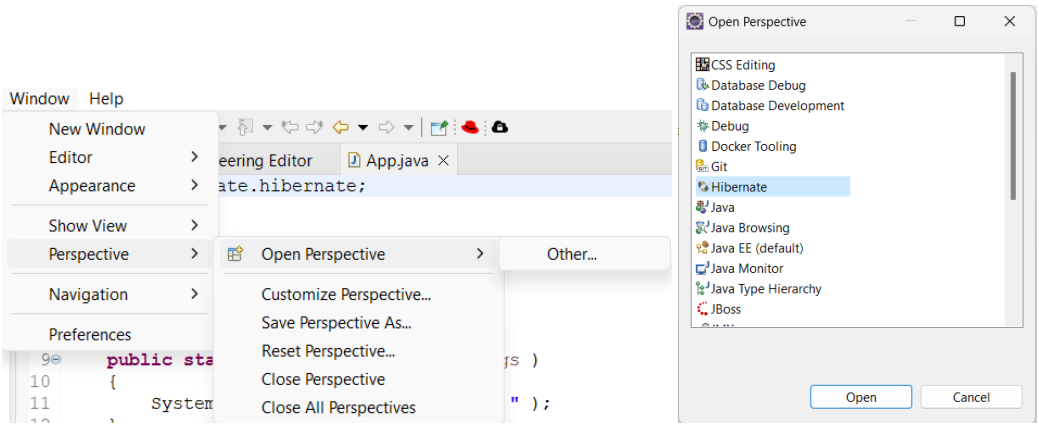
- **Seleccionas la carpeta del proyecto donde hemos guardado antes el archivo de configuración y si queremos podemos cambiarle el nombre. Le damos a Next.**



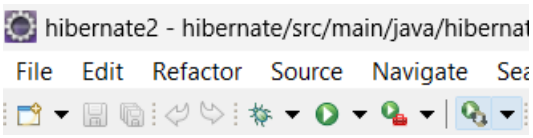
En el desplegable seleccionamos la consola que hemos creado antes y que le hemos puesto el nombre de ConsolaConfiguracionEmpresa a continuación le damos a Refresh y podremos ver que se nos conecta con la base de datos y podremos elegir las tablas que tenemos que en nuestro caso son clientes y productos que están dentro de empresa los seleccionamos y le damos a Include… , y por último le damos a Finish.



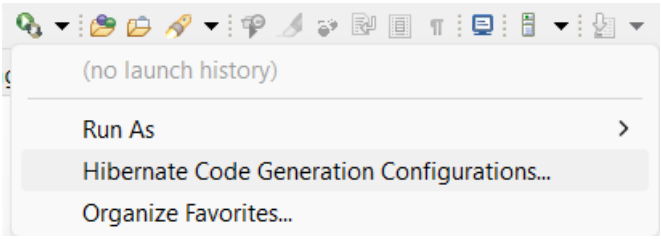
Al darle doble clic al nuevo archivo que hemos creado y después seleccionamos la vista Source podremos ver el código con los datos que le hemos dicho al crear el archivo.



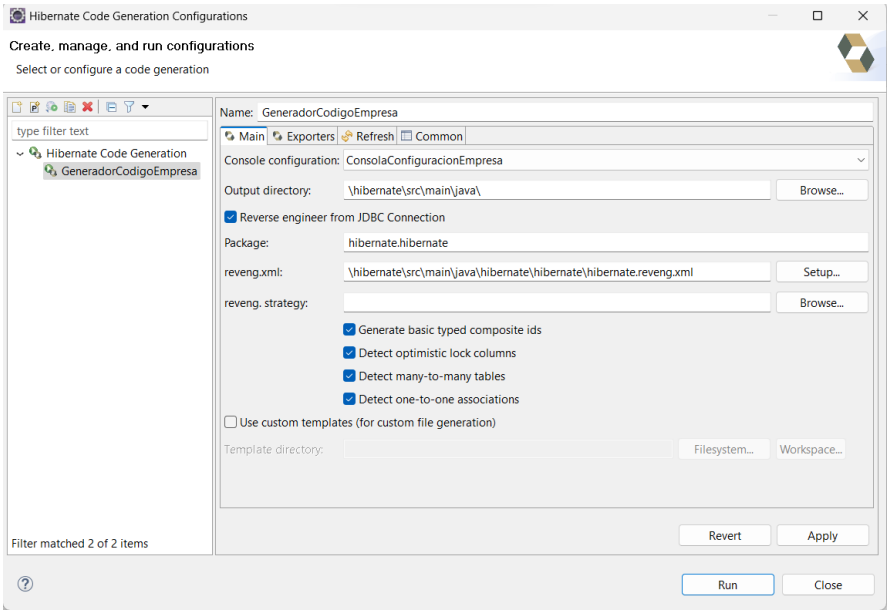
- **Tenemos que mostrar la perspectiva de Hibernate que se abre dando clic en Window > Perspective >Open Perspective > Other..**
  Una vez que se nos abre la ventana de las perspectivas buscamos Hibernate y hacemos clic en Open.



- **En la parte izquierda arriba del ide tendremos un nuevo icono Run, pero con el icono de Hibernate.**



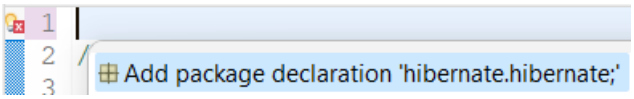
- **Le damos al icono y después a la opción de Hibernate Code Generation Configurations…**



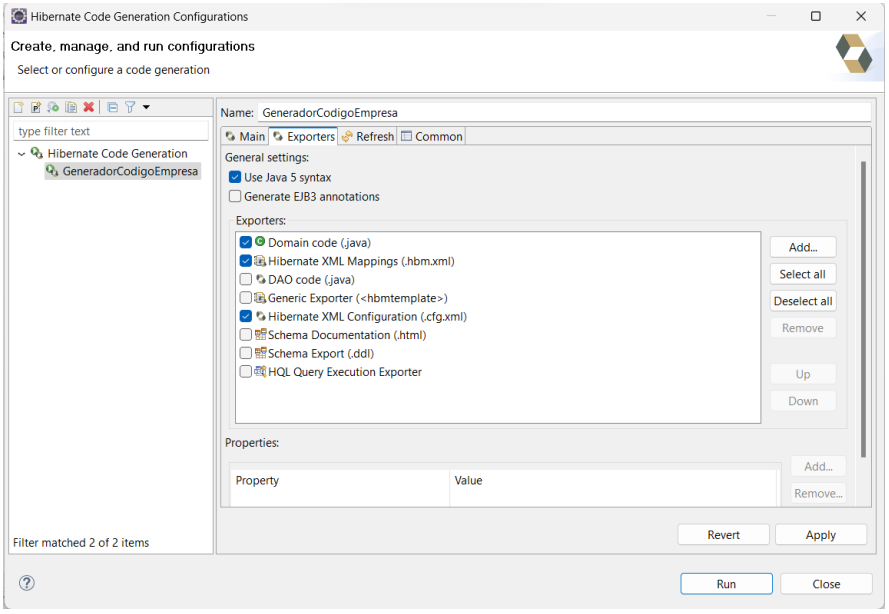
- **Se nos abrirá la siguiente ventana en la cual tendremos que darle doble clic en la parte izquierda sobre Hibernate Code Generation para que se nos cree una configuración nueva después deberemos ponerle un nombre que en este caso sería GeneradorCodigoEmpresa.**
- **A continuación, los datos de la configuración de consola que en nuestro caso sería ConsolaConfiguracionEmpresa, directorio que en este caso sería \hibernate\src\main\java\ que es donde está el paquete con las clases, en Package el nombre de nuestro paquete que es hibernate.hibernate, en el tenemos creados los**

anteriores archivos de configuración de Hibernate y por último el archivo de reveng.xml que contendrá la información sobre las tablas y que está en el mismo paquete, eso si al darle a Browse… le diremos que un archivo que ya existe y a continuación lo buscaremos.

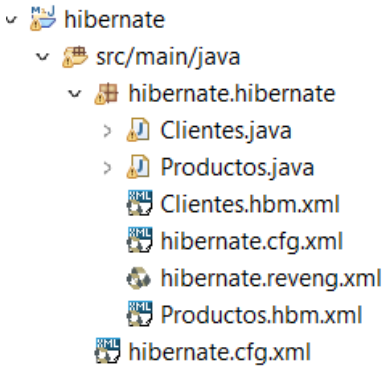
- *Si ponemos toda la ruta en el directorio las clases nos salen en rojo porque al crearlas no les pone el paquete en el que se encuentran, pero si le damos al warning lo podemos agregar nosotros.



- Vamos a la siguiente pestaña que es Exporters para configurar lo siguiente.



- Marcamos las opciones Use Java 5 sytax, Domain code(.java),Hibernate XML Mappings (.hbm.xml), Hibernate XML Configuration (.cfg.xml). Después hacemos clic en Apply y por último a Run.



- Podremos observar que se nos han creado los siguientes ficheros con la información de las tablas de la base de datos y aparte otro fichero de configuración que nos

servirá después cuando creamos las clases para hacer operaciones con la base de datos.

```java
package hibernate.hibernate;
// Generated 13 ene 2024, 15:53:48 by Hibernate Tools 6.3.1.Final

/**
 * Clientes generated by hbm2java
 */
public class Clientes implements java.io.Serializable {

    private Integer id;
    private String nombre;
    private String pais;

    public Clientes() {
    }

    public Clientes(String nombre, String pais) {
        this.nombre = nombre;
        this.pais = pais;
    }

    public Integer getId() {
        return this.id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

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

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

```java
package hibernate.hibernate;
// Generated 13 ene 2024, 15:53:48 by Hibernate Tools 6.3.1.Final

/**
 * Productos generated by hbm2java
 */
public class Productos implements java.io.Serializable {

    private Integer id;
    private String nombre;
    private double precio;

    public Productos() {
    }

    public Productos(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }

    public Integer getId() {
        return this.id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

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

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

hibernate.cfg.xml

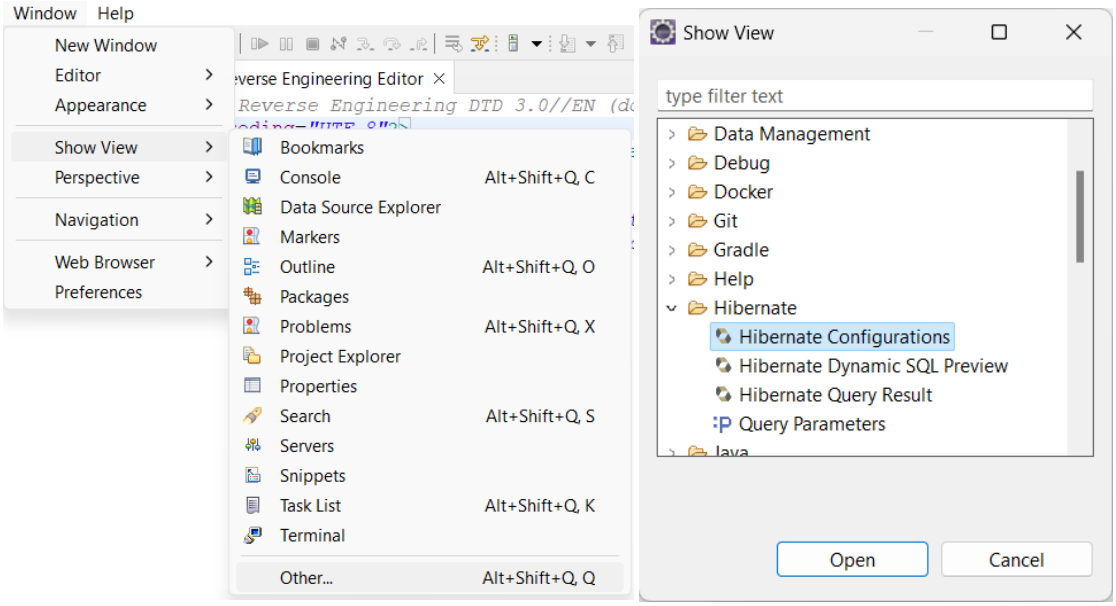
```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "ht
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost/empresa</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.hbm2ddl.auto">none</property>
        <property name="hibernate.search.autoregister_listeners">true</property>
        <property name="hibernate.validator.apply_to_ddl">false</property>
        <mapping resource="hibernate/hibernate/Productos.hbm.xml"/>
        <mapping resource="hibernate/hibernate/Clientes.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

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

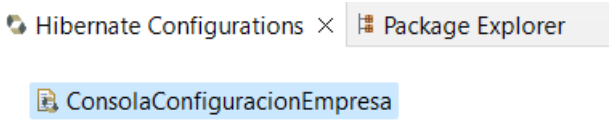
```
-//Hibernate/Hibernate Mapping DTD 3.0//EN (doctype with catalog)
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Generated 13 ene 2024, 15:53:48 by Hibernate Tools 6.3.1.Final --><!DOCTYPE hibernate-mapping P
<hibernate-mapping>
    <class catalog="empresa" name="hibernate.hibernate.Clientes" optimistic-lock="none" table="clien
        <id name="id" type="java.lang.Integer">
            <column name="id"/>
            <generator class="identity"/>
        </id>
        <property name="nombre" type="string">
            <column length="20" name="nombre" not-null="true"/>
        </property>
        <property name="pais" type="string">
            <column length="15" name="pais" not-null="true"/>
        </property>
    </class>
</hibernate-mapping>
```

```
-//Hibernate/Hibernate Mapping DTD 3.0//EN (doctype with catalog)
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Generated 13 ene 2024, 15:53:48 by Hibernate Tools 6.3.1.Final --><!DOCTYPE hibernate-mapping P
<hibernate-mapping>
    <class catalog="empresa" name="hibernate.hibernate.Productos" optimistic-lock="none" table="proc
        <id name="id" type="java.lang.Integer">
            <column name="id"/>
            <generator class="identity"/>
        </id>
        <property name="nombre" type="string">
            <column length="20" name="nombre" not-null="true"/>
        </property>
        <property name="precio" type="double">
            <column name="precio" not-null="true" precision="22" scale="0"/>
        </property>
    </class>
</hibernate-mapping>
```

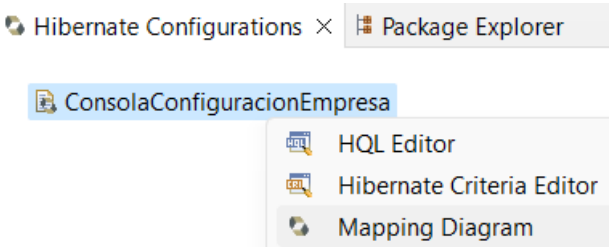
- **Los archivos de Clientes.hbm.xml y Productos.hbm.xml con información sobre el mapeo de las clases con las tablas.**



- **Debemos abrirnos la vista de Hibernate Configurations haciendo clic en la pestaña Window > Show View > Other > Hibernate > Hibernate Configurations.**

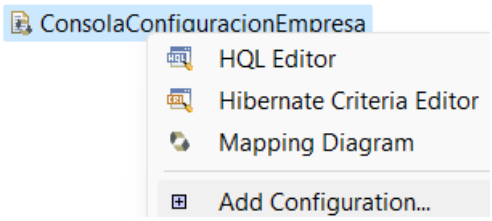


- **En la parte izquierda arriba se nos abrirá otra pestaña llamada Hibernate Configurations.**

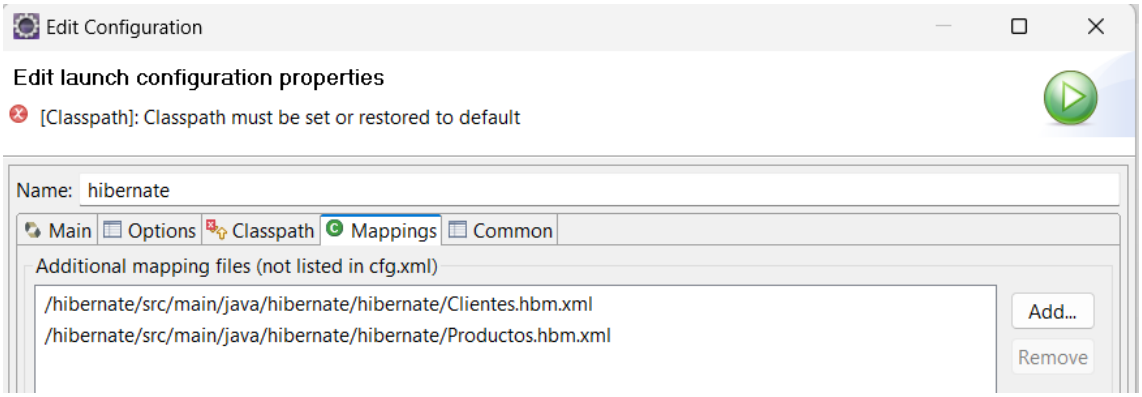


- **Clic derecho sobre la ConsolaConfiguracionEmpresa > Mapping Diagram.**

- **Este seria el diagrama que se ha creado sobre nuestras tablas de la base de datos que como podremos observar ha funcionado correctamente.**
- **Si da un error en rojo, una de las soluciones fue la siguiente:**



- **Clic derecho sobre Consola > Add Configuration…**



- **Vamos a la pestaña de Mappings y ponemos los archivos de las clases.hbm.xml y volvemos a probar con Mapping Diagram.**

Buscamos las dependencias de Hibernate ORM Hibernate Core última versión.

https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-core

Hibernate ORM Hibernate Core » 6.4.1.Final

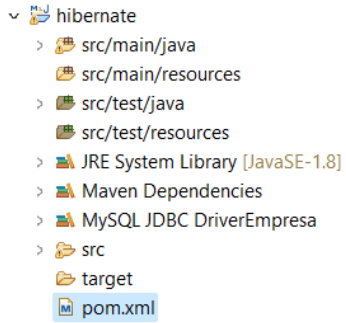
Hibernate's core ORM functionality

License	LGPL 2.1
Categories	Object/Relational Mapping
Tags	persistence mapping orm hibernate relational
Organization	Hibernate.org
HomePage	https://hibernate.org/orm
Date	Dec 15, 2023
Files	pom (5 KB) jar (11.0 MB) View All
Repositories	Central
Ranking	#2021 in MvnRepository (See Top Artifacts) #6 in Object/Relational Mapping
Used By	226 artifacts

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

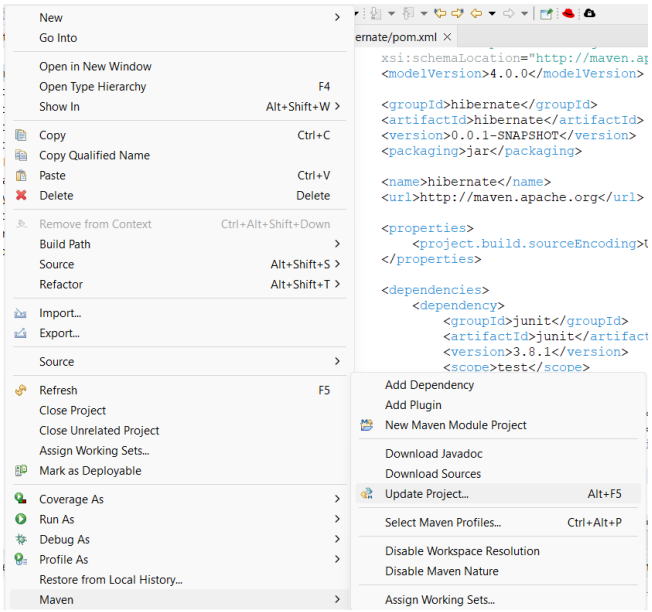
```
<!-- https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-core -->
<dependency>
    <groupId>org.hibernate.orm</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>6.4.1.Final</version>
</dependency>
```

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

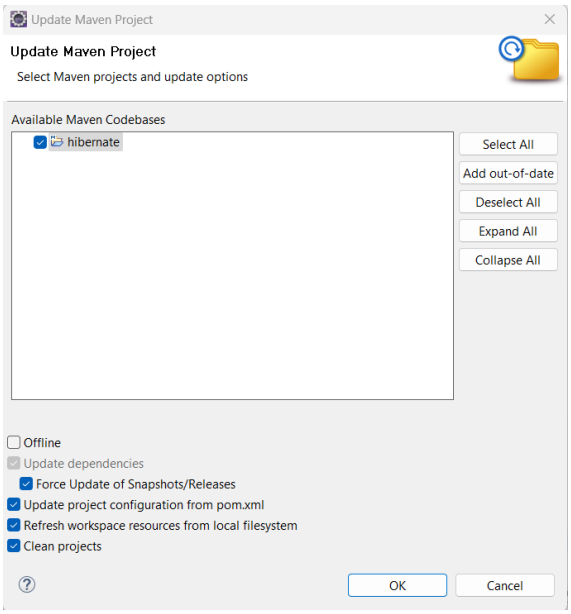


```
<dependencies>
  <dependency>
      <groupId>org.hibernate.orm</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>6.4.1.Final</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

- **¡CUIDADO! ¡Poner la de Hibernate primero, de lo contrario es posible que no funcione!**

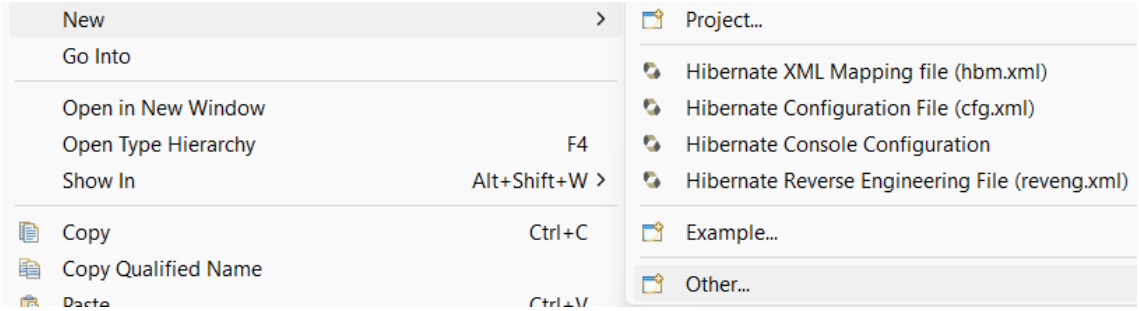


- **Clic derecho sobre el proyecto > Maven > Update Project.**

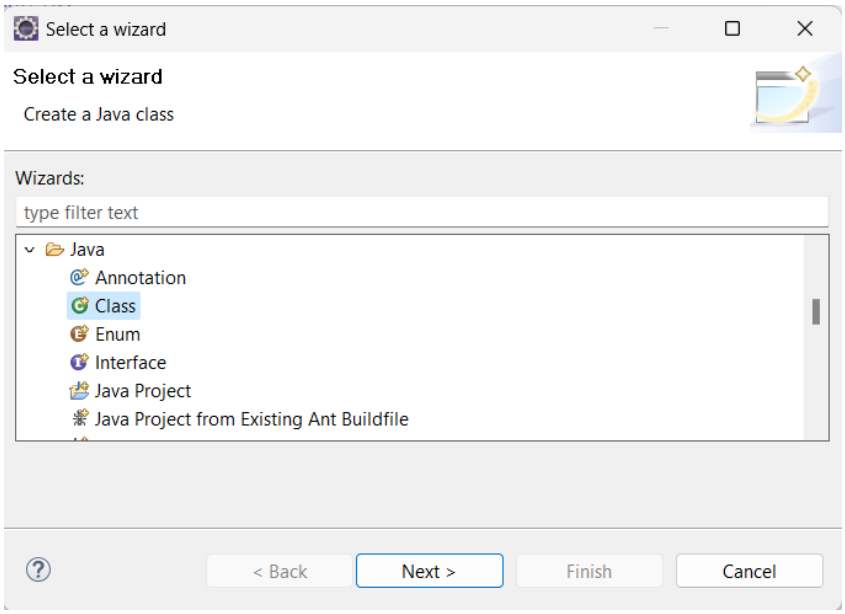


Le damos a la opción Force Update of Snapshots/Releases y luego a OK. Para que se actualicen las dependencias y bajen la nueva dependencia.

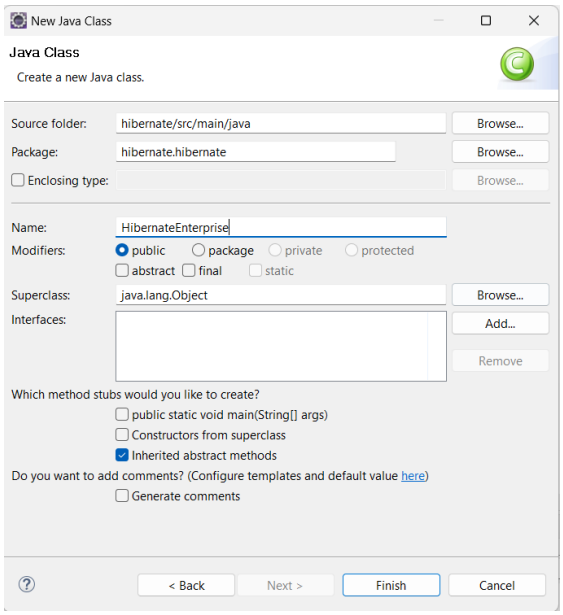
Usar Hibernate para hacer operaciones en la Base de Datos.



Clic derecho sobre el paquete del proyecto >New > Other…



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



Le ponemos el nombre de HibernateEnterprise y le damos a Finish.

```
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 {
```

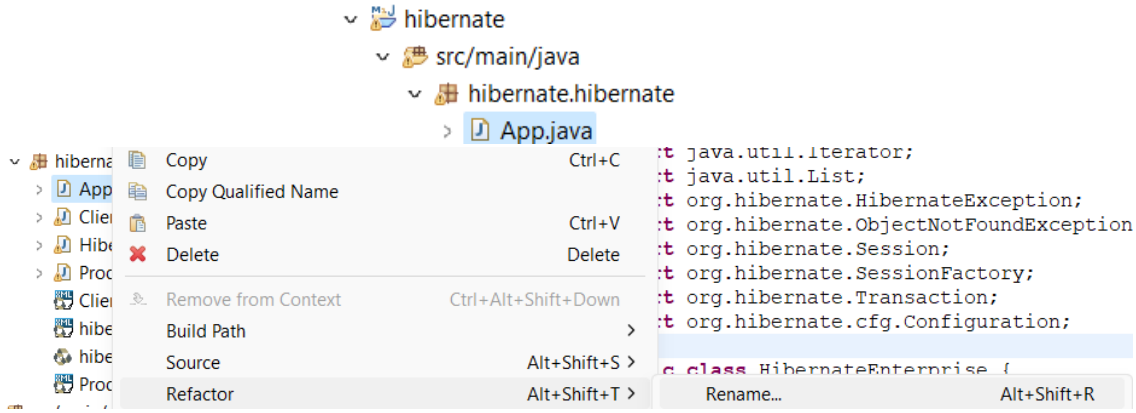
```
	private static SessionFactory sf; // this SessionFactory will
be created once and used for all the connections
	private static Productos p;

	HibernateEnterprise() {// constructor
		// sf = HibernateUtil.getSessionFactory();
		sf = new
Configuration().configure().buildSessionFactory(); // also works
	}

	public void close() {
		sf.close();
	}

	public void addProduct(String name, double price) {
		Session session = sf.openSession();// session es la
variable que tiene el método
                                                                //
save para guardar productos

		Transaction tx = null;
		// create the product with the parameters in the method
		Productos p = new Productos();
		p.setNombre(name);
		p.setPrecio(price);
		// 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, price);

	System.out.println("====================================");
			tx = session.beginTransaction();
			session.save(p);// we INSERT p into the table
PRODUCTS
			tx.commit();// if session.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 {
			session.close();
		}
	}

	public void showProducts() {
		Session session = sf.openSession();
		Transaction tx = null;

		try {
			tx = session.beginTransaction();
			List allproducts = session.createQuery("From
Productos").list();

			Iterator it = allproducts.iterator();

	System.out.println("====================================");
```

```
                  System.out.println("Buscando Productos...");

      System.out.println("=====================================");
                  while (it.hasNext()) {
                        // for (Iterator 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 {
                  session.close();
            }
      }

      public Productos findProductById(int id) {
            Session session = 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 = session.beginTransaction();
                  p = (Productos) session.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) {
```

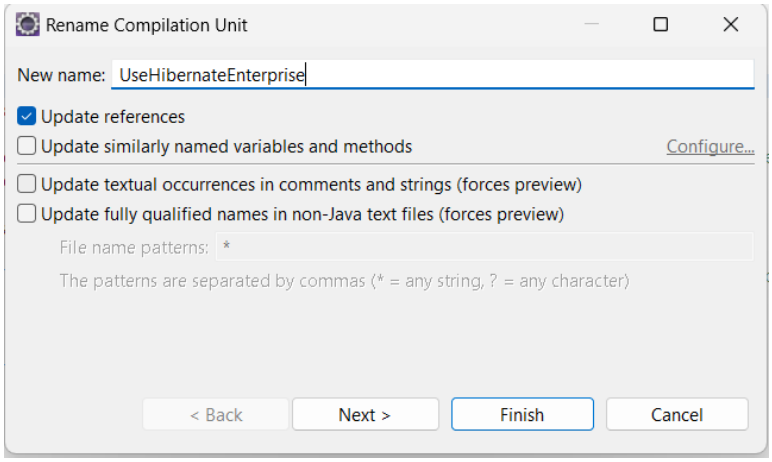
```
                        System.out.println(e);
                        tx.rollback();
                  }
            } finally {
                  session.close();
            }
            return p;
      }

      public void deleteProductById(int id) {
            Productos p = new Productos();
            Session session = sf.openSession();
            Transaction tx = null;
            try {

      System.out.println("=====================================");
                  System.out.println("Buscando Producto con ID -> " +
id);

      System.out.println("=====================================");
                  tx = session.beginTransaction();
                  p = (Productos) session.get(Productos.class, id);

                  if (p != null) {

      System.out.println("=====================================");
                        System.out.println("Borrando Producto de la
Base de Datos...");

      System.out.println("=====================================");

                        session.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 {
                  session.close();
            }
      }
```

```
	public void updateProductById(int id, String newName, double
newPrice) {
		Productos p = new Productos();
		Session session = 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 = session.beginTransaction();
			p = (Productos) session.load(Productos.class,
id);// we load the product

	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);
			session.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 {
			session.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.**

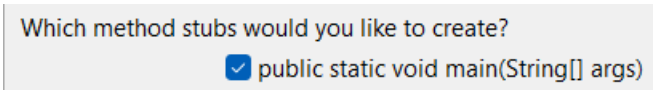


- **Se nos ha creado una clase Main por defecto llamada App.Java, le damos clic derecho sobre la clase > Refactor > Rename…**



- **Le ponemos el nombre UseHibernateEnterprise. Le damos a Next, de nuevo a Next y Finish.**

- **¡Podemos Borrar la que viene por defecto! Y crear una nueva igual que hemos creado la anterior, pero hay que marcar esta opción.**



- **Creamos una nueva clase para probar los métodos de HibernateEnterprise y para poder probarlos.**

```
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",155);
//		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.**

```
==========================================
Buscando Productos...
==========================================
==========================================
Id: 1
Nombre: disco ssd
Precio: 50.0
==========================================
==========================================
Id: 2
Nombre: memoria DDR4
Precio: 35.5
==========================================
==========================================
Id: 8
Nombre: ssd
Precio: 155.0
==========================================
==========================================
Id: 9
Nombre: monitor
Precio: 170.0
==========================================
==========================================
Id: 10
Nombre: monitor
Precio: 170.0
==========================================
==========================================
Id: 11
Nombre: monitor
Precio: 170.0
==========================================
==========================================
Finalizada la Busqueda...
==========================================
```

- **Según esta esté código deberíamos de poder ver los productos de la tabla Productos.**