



HIBERNATE



eclipse



HIBERNATE EN ECLIPSE

David Cristian Pirosca

Índice

Proyecto Hibernate en Eclipse	1
Base de Datos empresa.....	1
Instalamos Plugin JBoss Tools del Marketplace.....	1
Creamos un nuevo Proyecto Maven.....	3
Descargamos el Jar del Conecotor y Creamos una Conectividad con la Base de Datos.....	5
Creamos los archivos de Hibernate para poder hacer un mapeo de objetos	10
Buscamos las dependencias de Hibernate ORM Hibernate Core última versión.	25
Usar Hibernate para hacer operaciones en la Base de Datos.....	27



Proyecto Hibernate en Eclipse

Base de Datos empresa.

The screenshot shows the MySQL Workbench interface. At the top, there's a tree view of the 'empresa' database structure, which contains three tables: 'Nueva', 'clientes', and 'productos'. Below this, there are two tabs: 'Servidor: 127.0.0.1 > Base de datos: empresa > Tabla: clientes' and 'Servidor: 127.0.0.1 > Base de datos: empresa > Tabla: productos'. Each tab displays the table structure with columns, data types, and constraints. The 'clientes' table has columns: id (int(11)), nombre (varchar(20)), and pais (varchar(15)). The 'productos' table has columns: id (int(11)), nombre (varchar(20)), and precio (double).

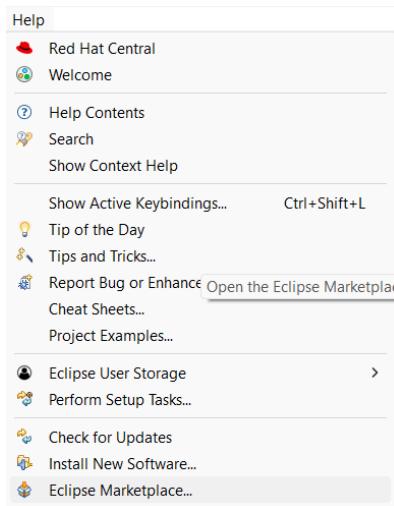
#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
2	nombre	varchar(20)	latin1_spanish_ci		No	Ninguna			Cambiar Eliminar Más
3	pais	varchar(15)	latin1_spanish_ci		No	Ninguna			Cambiar Eliminar Más

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
2	nombre	varchar(20)	latin1_spanish_ci		No	Ninguna			Cambiar Eliminar Más
3	precio	double			No	Ninguna			Cambiar Eliminar Más

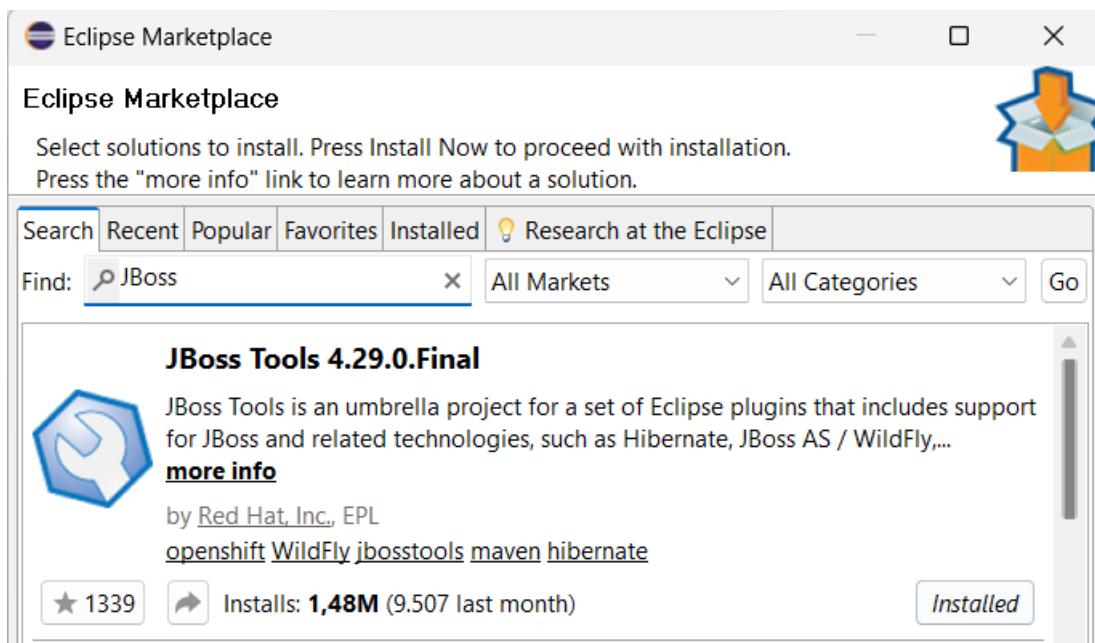
Descargar [empresa.sql](#)

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

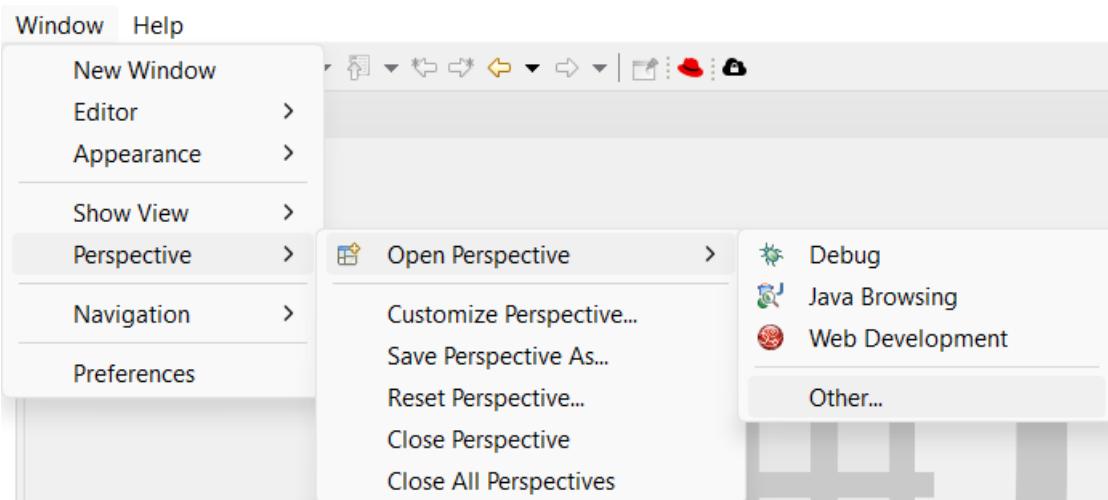
Instalamos Plugin JBoss Tools del Marketplace.



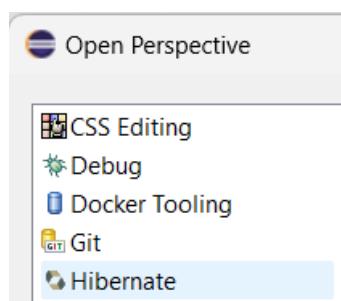
Help > Eclipse Marketplace



Buscamos JBoss y lo Instalamos con la configuración por Defecto.

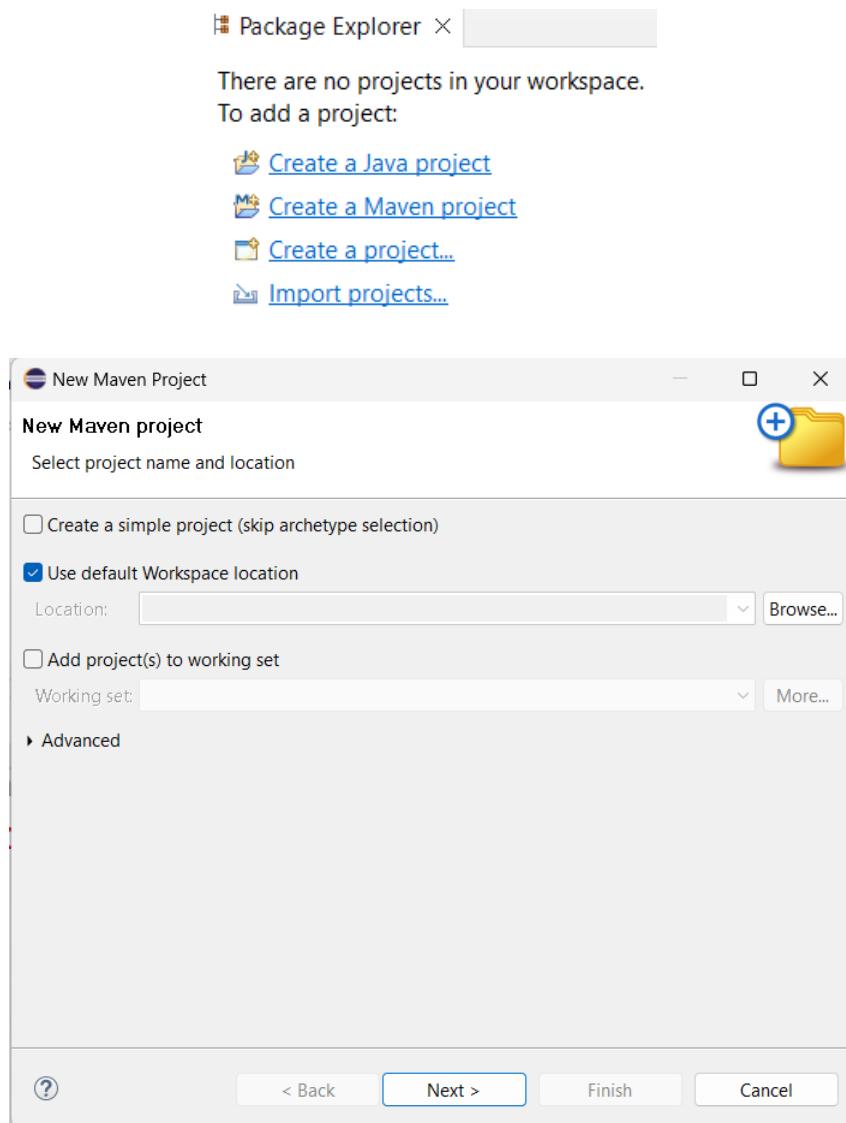


Comprobamos que está instalado Hibernate en Window > Perspective > Open Perspective > Other...



Podremos observar que está instalado.

Creamos un nuevo Proyecto Maven.

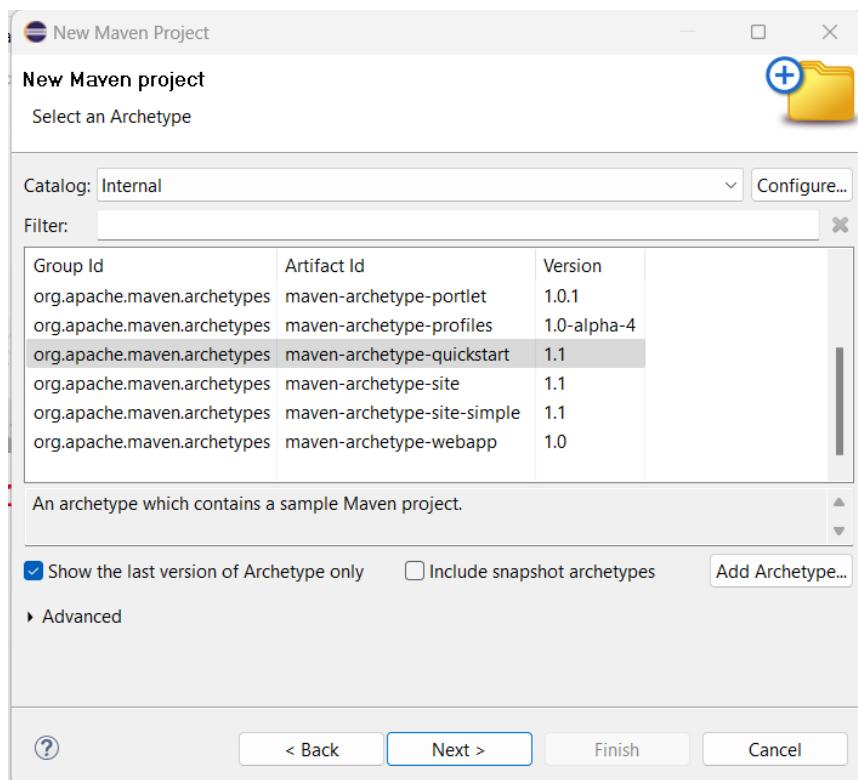


En este caso lo guardamos en la ubicación del Espacio de Trabajo si quisieramos otra ubicación lo podríamos cambiar y le damos a next.

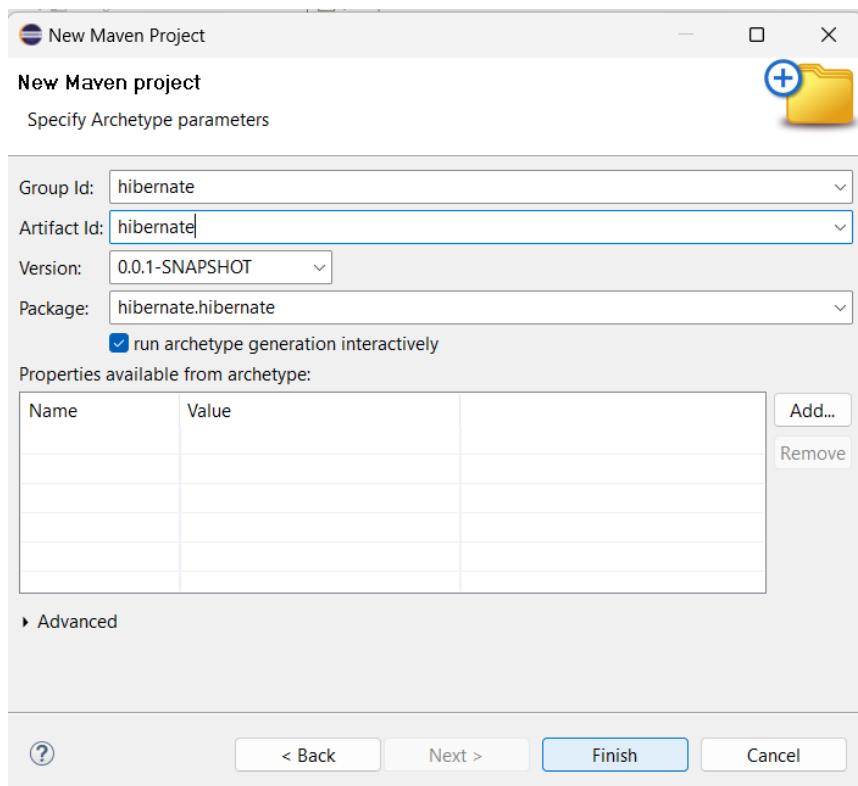
Hibernate

Eclipse

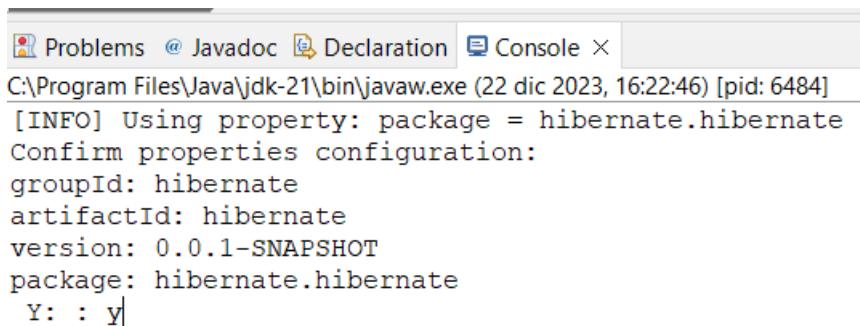
David Cristian Piroscia



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



Le ponemos nombre y le damos a finish.



```
C:\Program Files\Java\jdk-21\bin\javaw.exe (22 dic 2023, 16:22:46) [pid: 6484]
[INFO] Using property: package = hibernate.hibernate
Confirm properties configuration:
groupId: hibernate
artifactId: hibernate
version: 0.0.1-SNAPSHOT
package: hibernate.hibernate
Y: : y
```

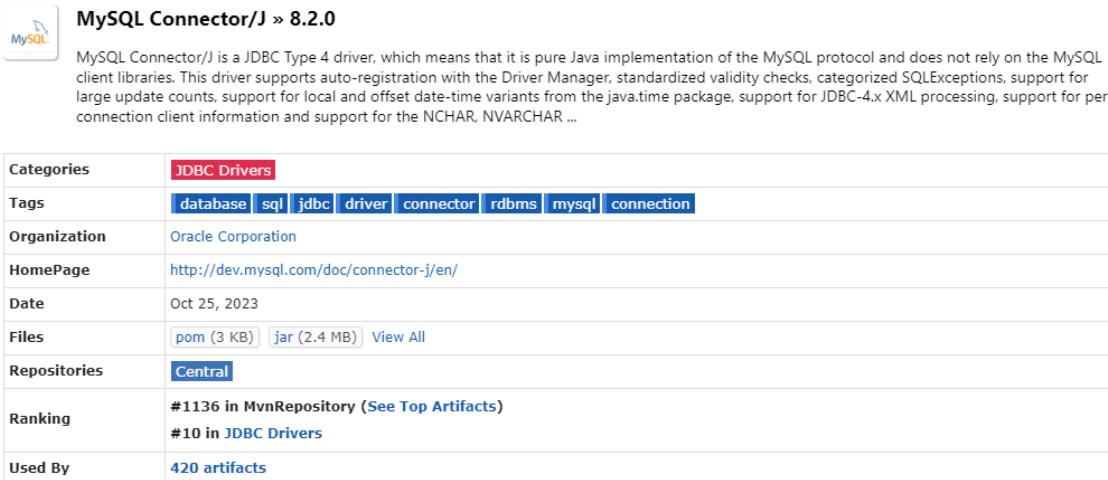
En la consola nos piden confirmación para la creación ponemos “y” para poder seguir.

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  25.443 s
[INFO] Finished at: 2023-12-22T16:23:13+01:00
[INFO] -----
```

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

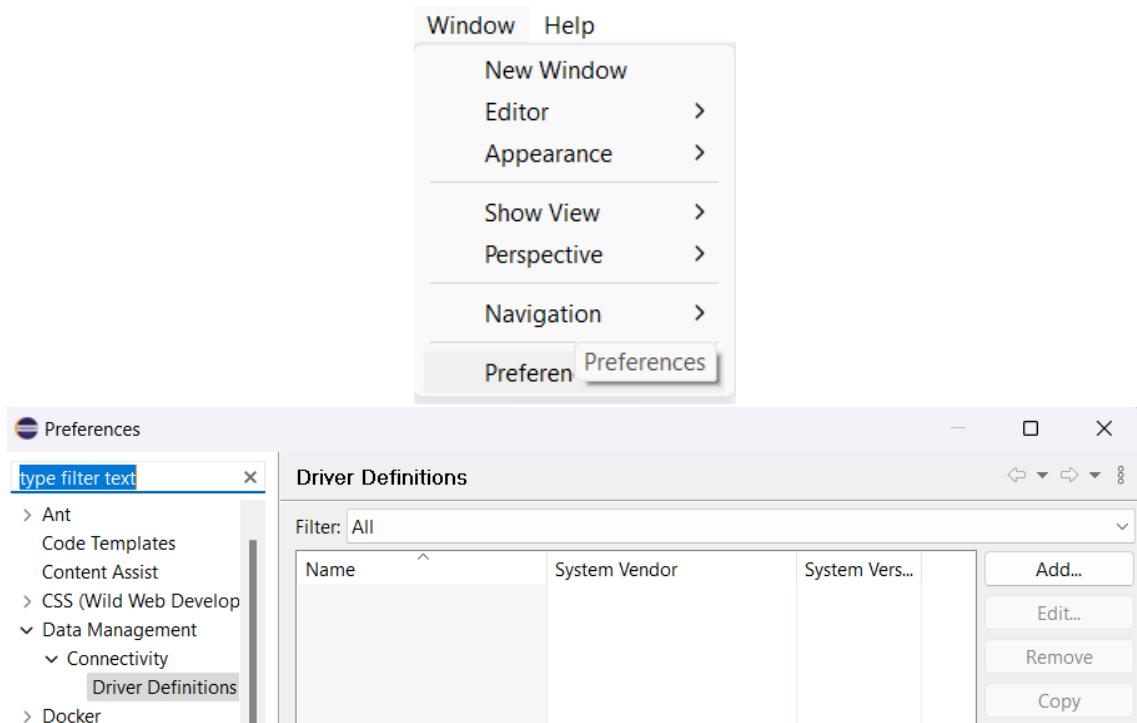
Descargamos el Jar del Conector y Creamos una Conectividad con la Base de Datos.

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



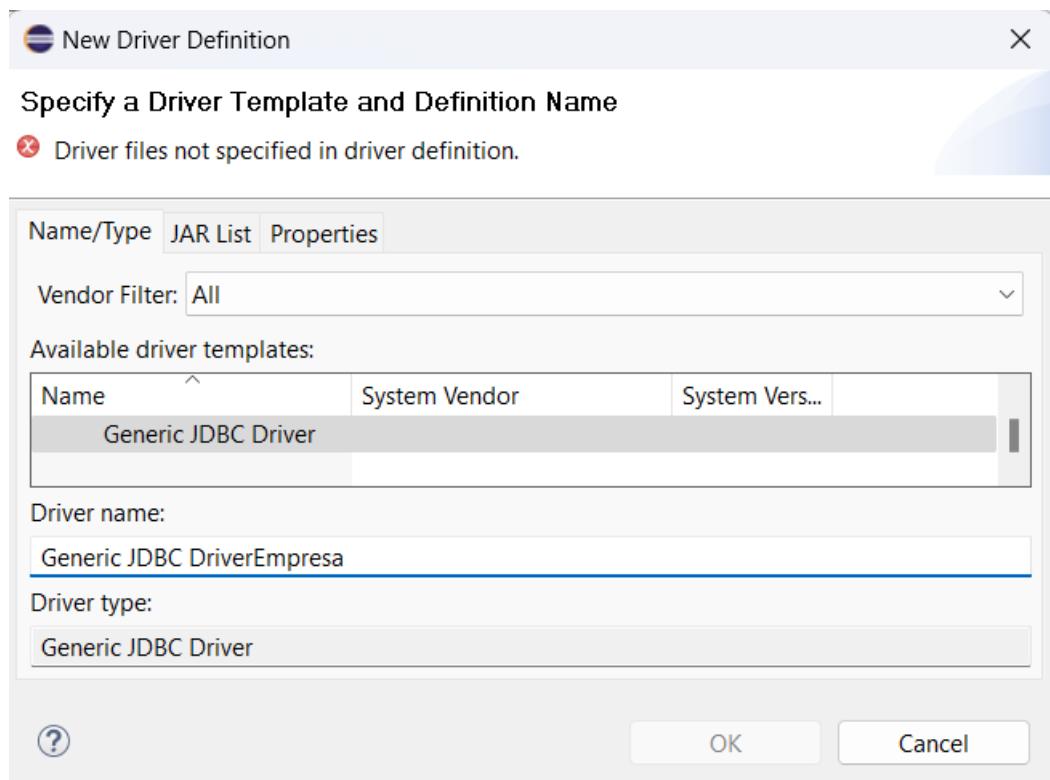
	MySQL Connector/J » 8.2.0
MySQL Connector/J is a JDBC Type 4 driver, which means that it is pure Java implementation of the MySQL protocol and does not rely on the MySQL client libraries. This driver supports auto-registration with the Driver Manager, standardized validity checks, categorized SQLExceptions, 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.

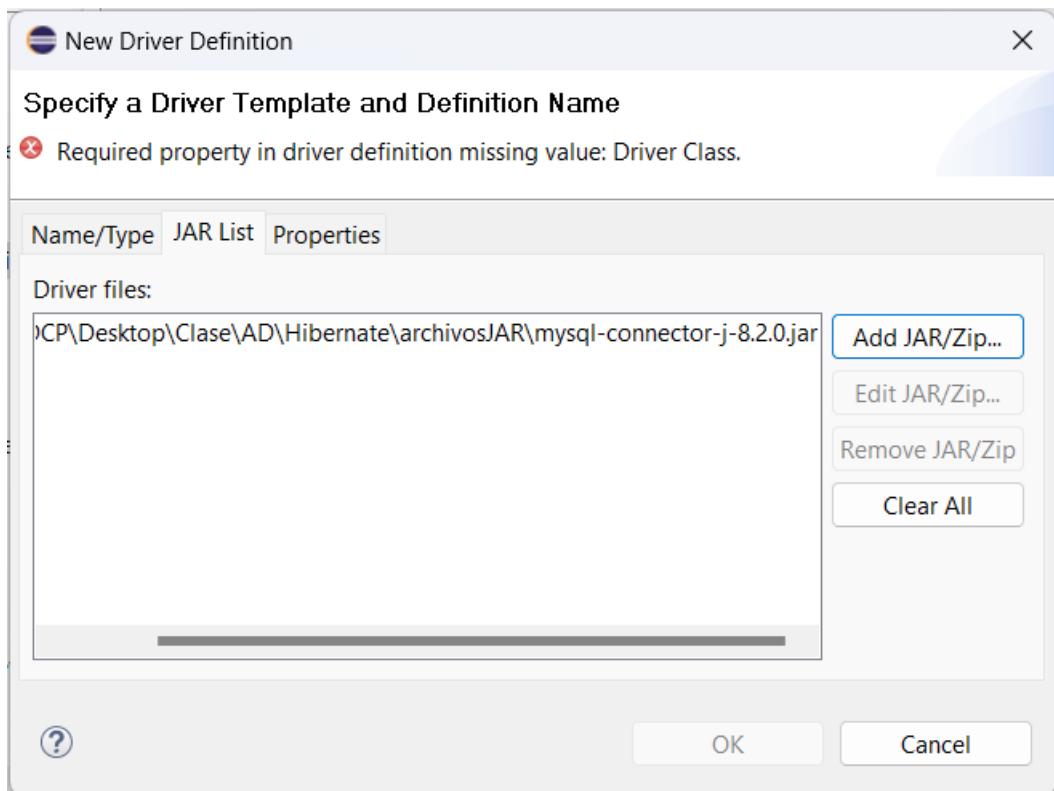


Vamos a Window > Preference > Data Management > Conectivity > Add..

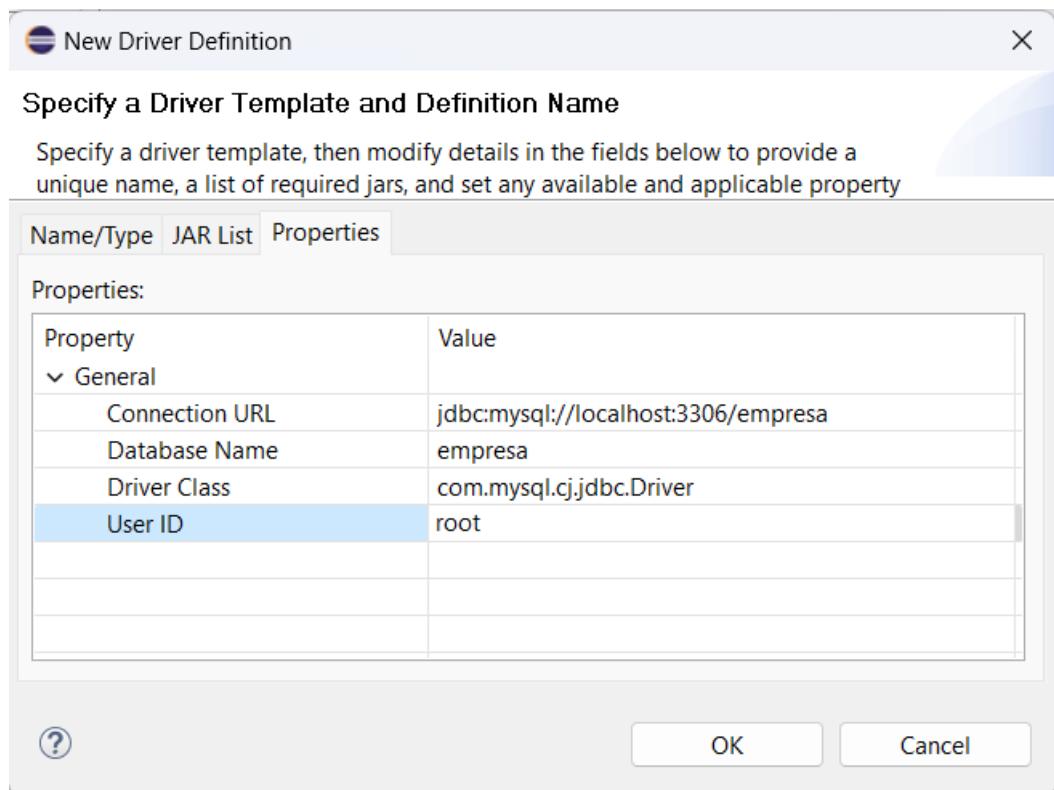
Para agregar una nueva conectividad a la base de datos.



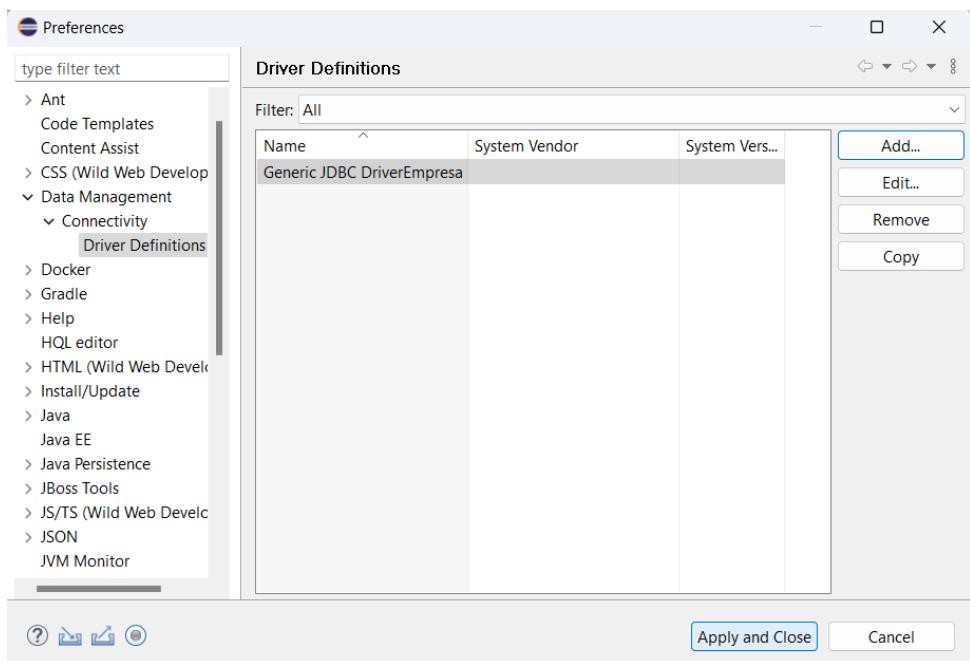
Seleccionamos el Driver y le ponemos el nombre que queramos en este caso se le ha puesto Empresa para saber que la conexión se hará sobre la base de datos empresa.



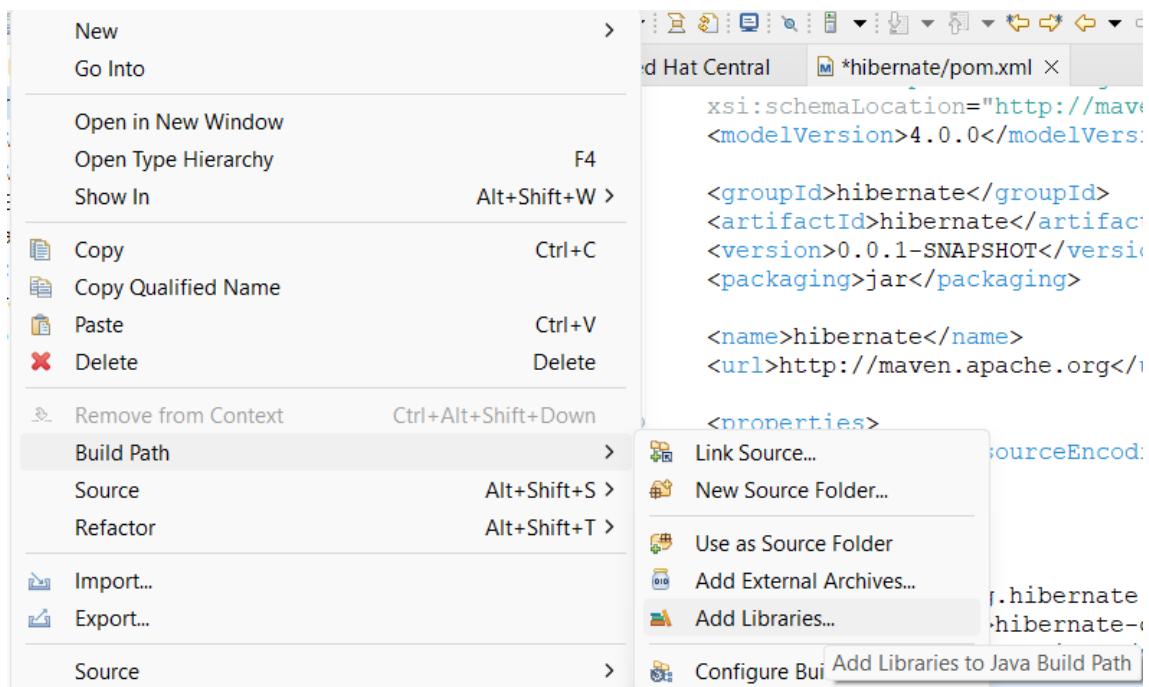
En la siguiente pestaña tendremos que poner la JAR del conector que acabamos de descargarnos.



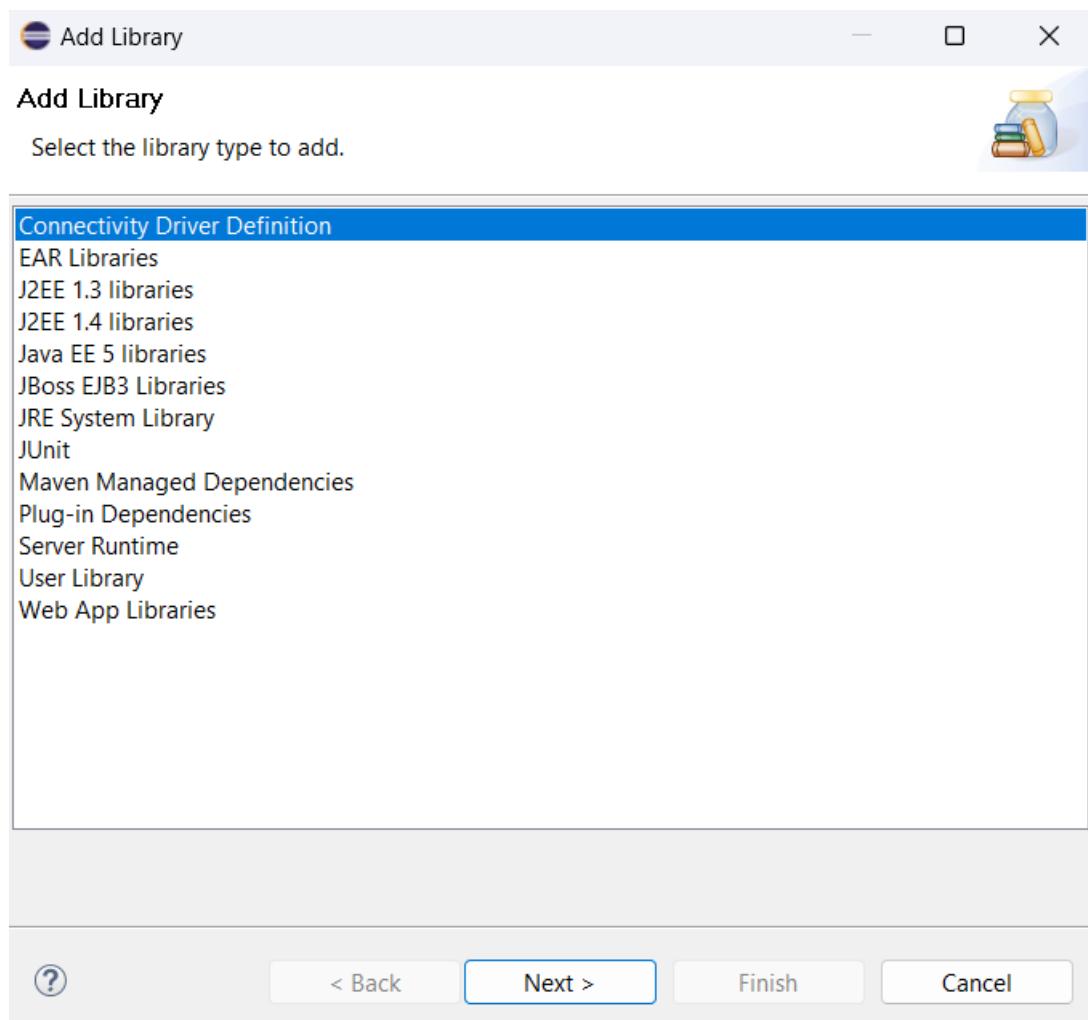
En la siguiente pestaña tendremos que poner los datos de la base de datos que nos vamos a conectar. Y el Driver que este caso es el más nuevo.



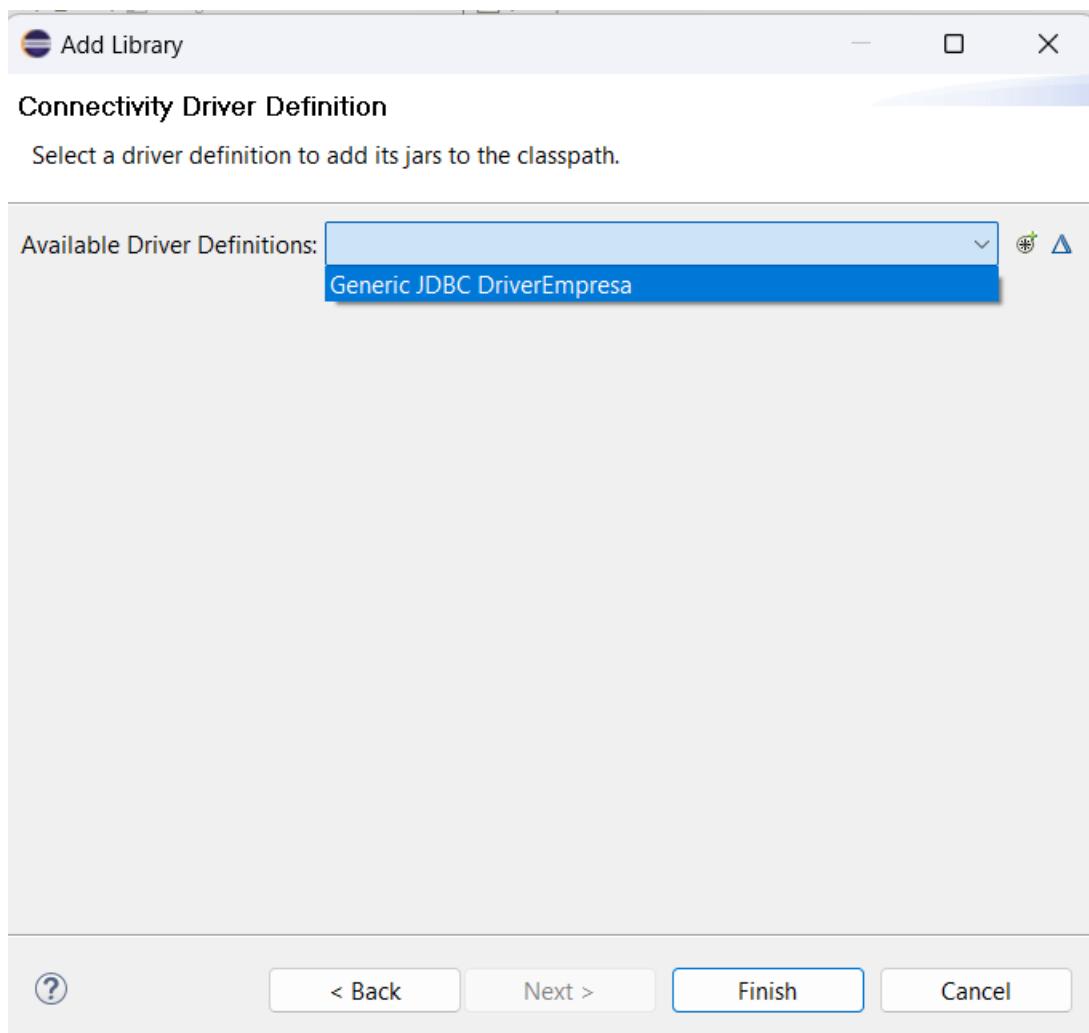
Aceptamos y le damos a aplicar y cerrar.



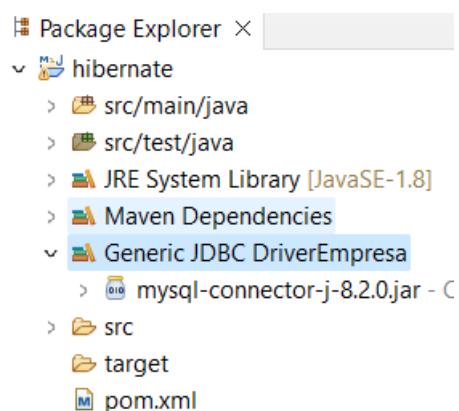
Hacemos clic derecho sobre el proyecto > Build Path > Add Libraries...



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

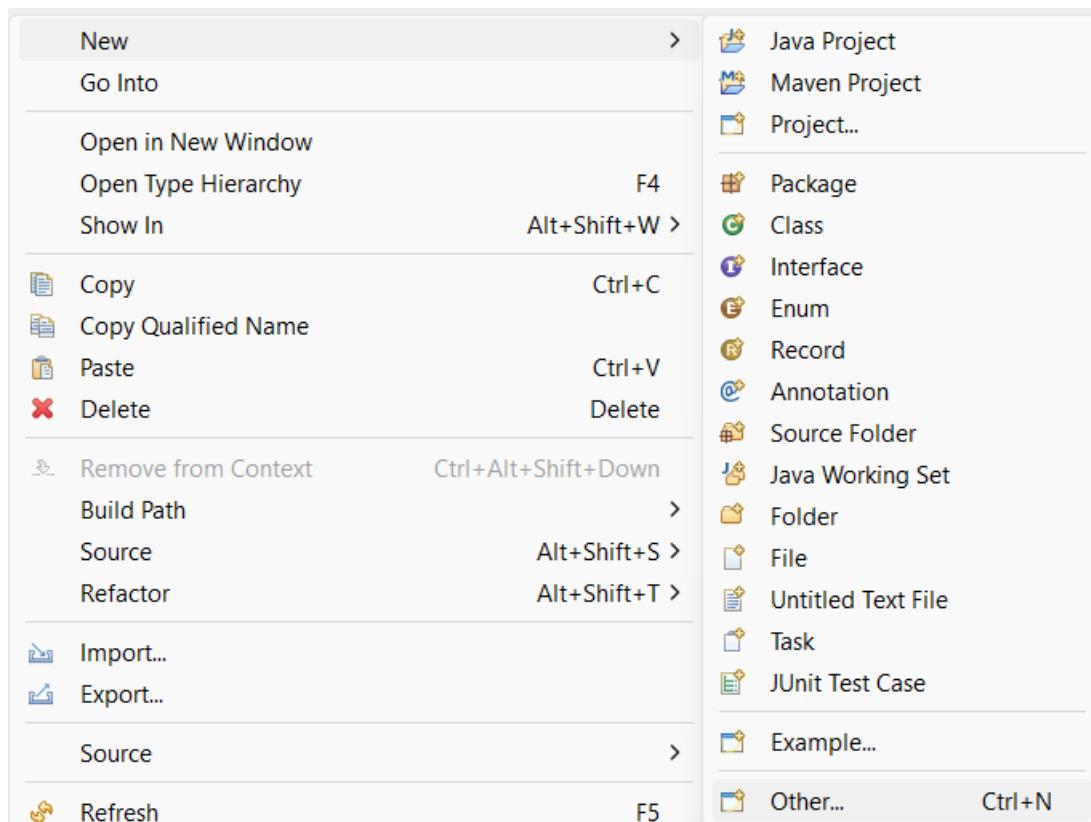


En la siguiente ventana le damos al desplegable y elegimos lo que acabamos de crear que nos aparece, y por último le damos a Finish.

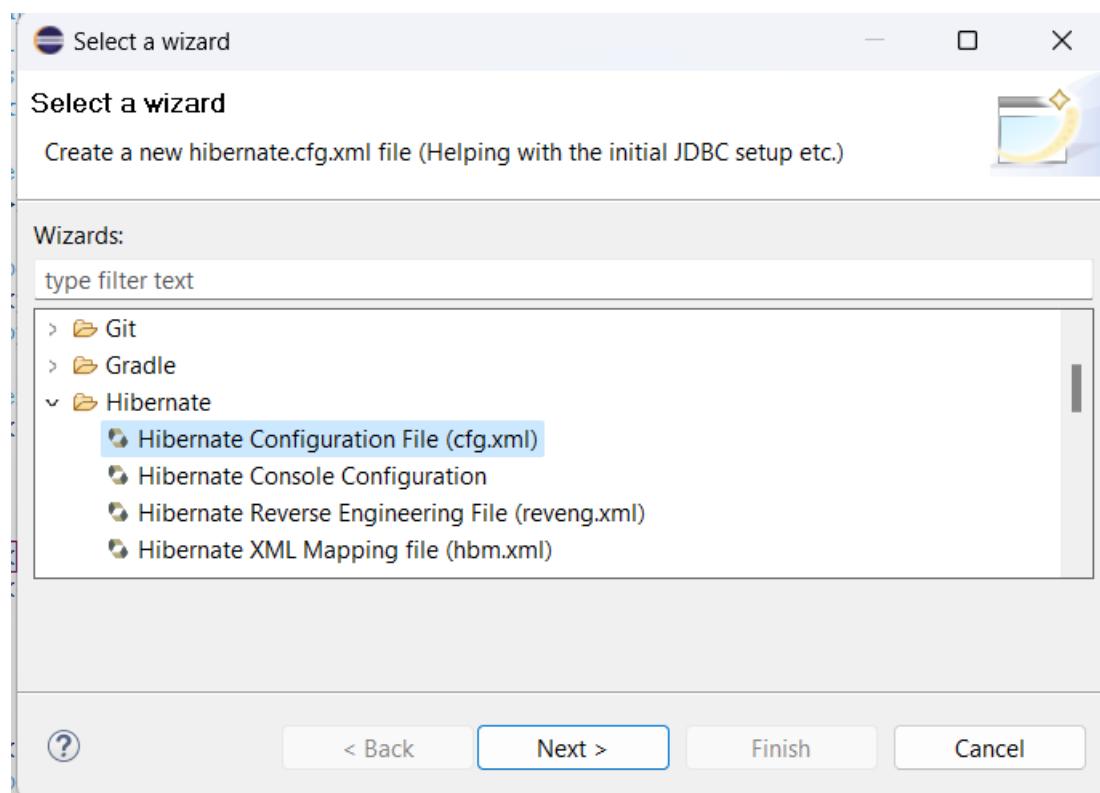


Como podremos observar en el proyecto ya lo tenemos.

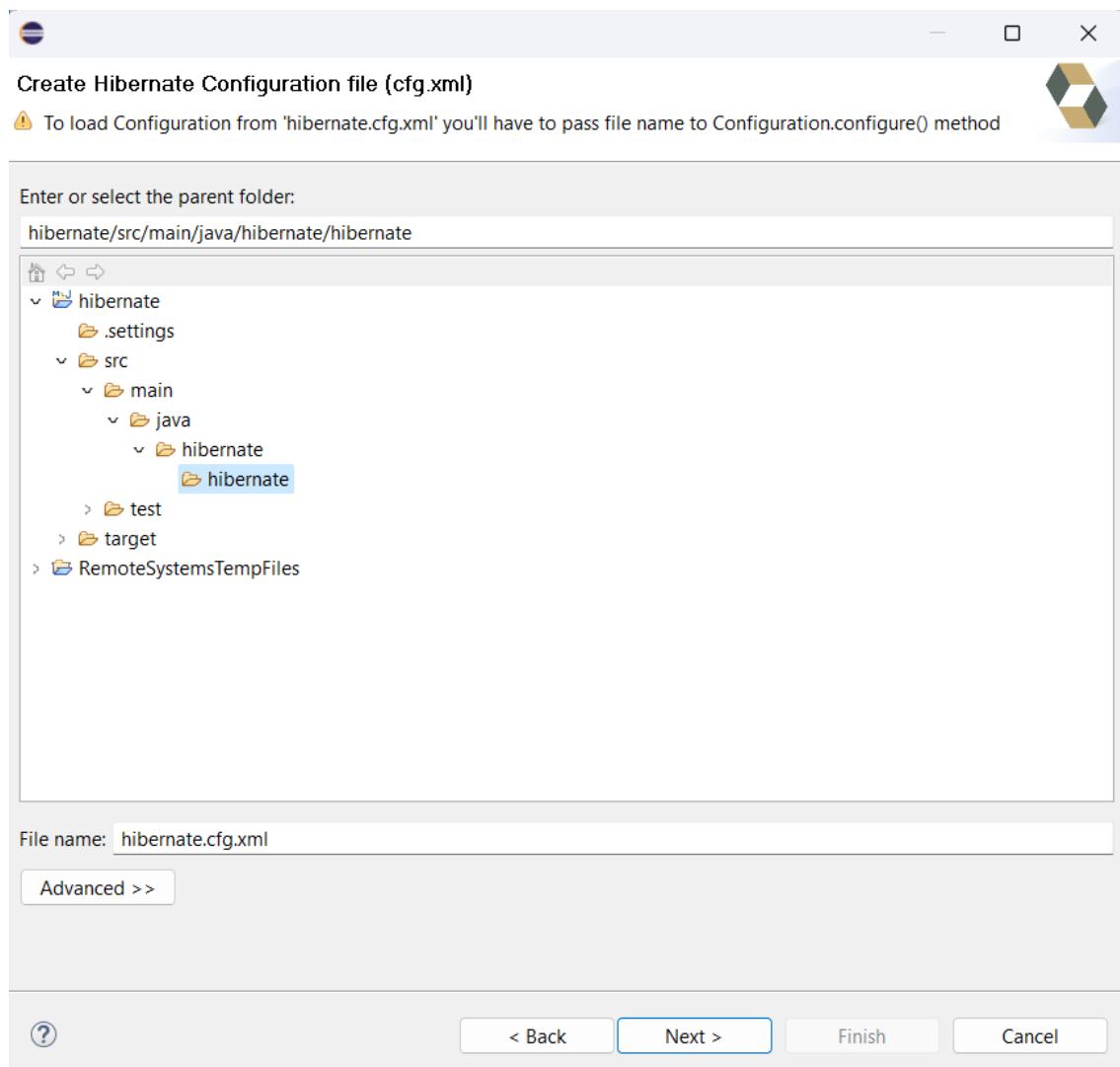
Creamos los archivos de Hibernate para poder hacer un mapeo de objetos



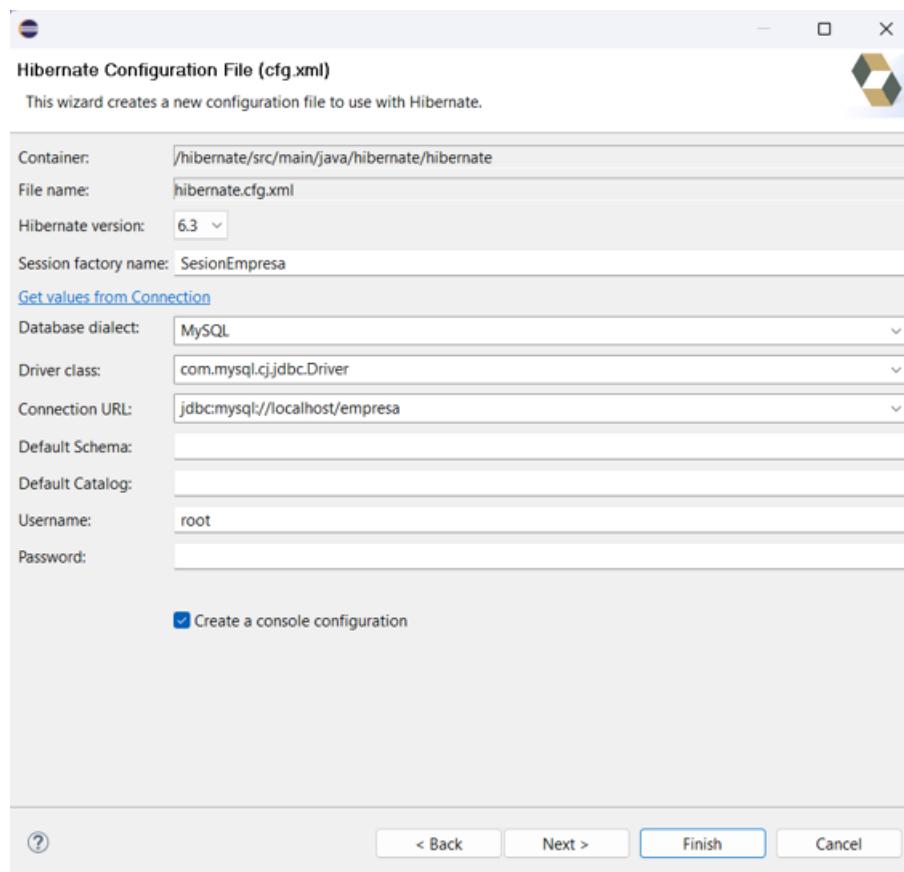
Ahora vamos a crear y configurar el archivo de Configuración de Hibernate. Que esta en New > Other...



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.



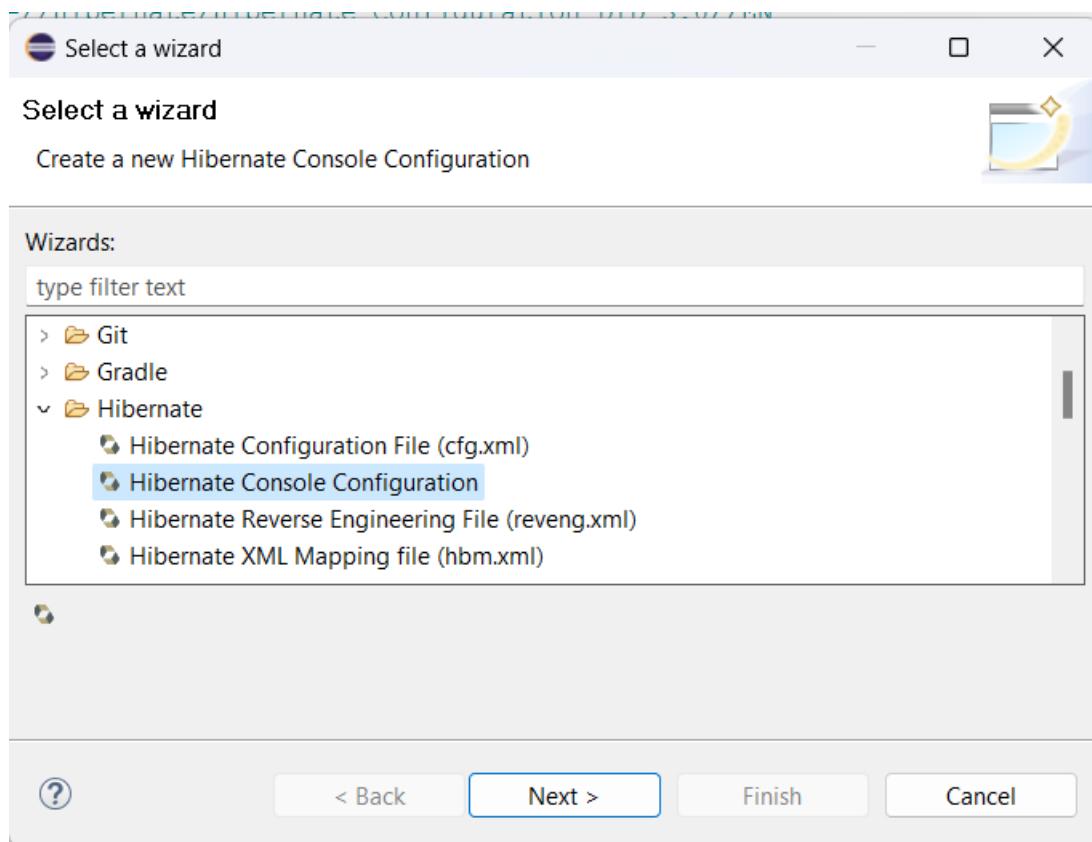
Rellenamos con la configuración de la base de datos y le ponemos un nombre y le damos al Check de Create a console configuration. Por último, le damos a Finish.

```

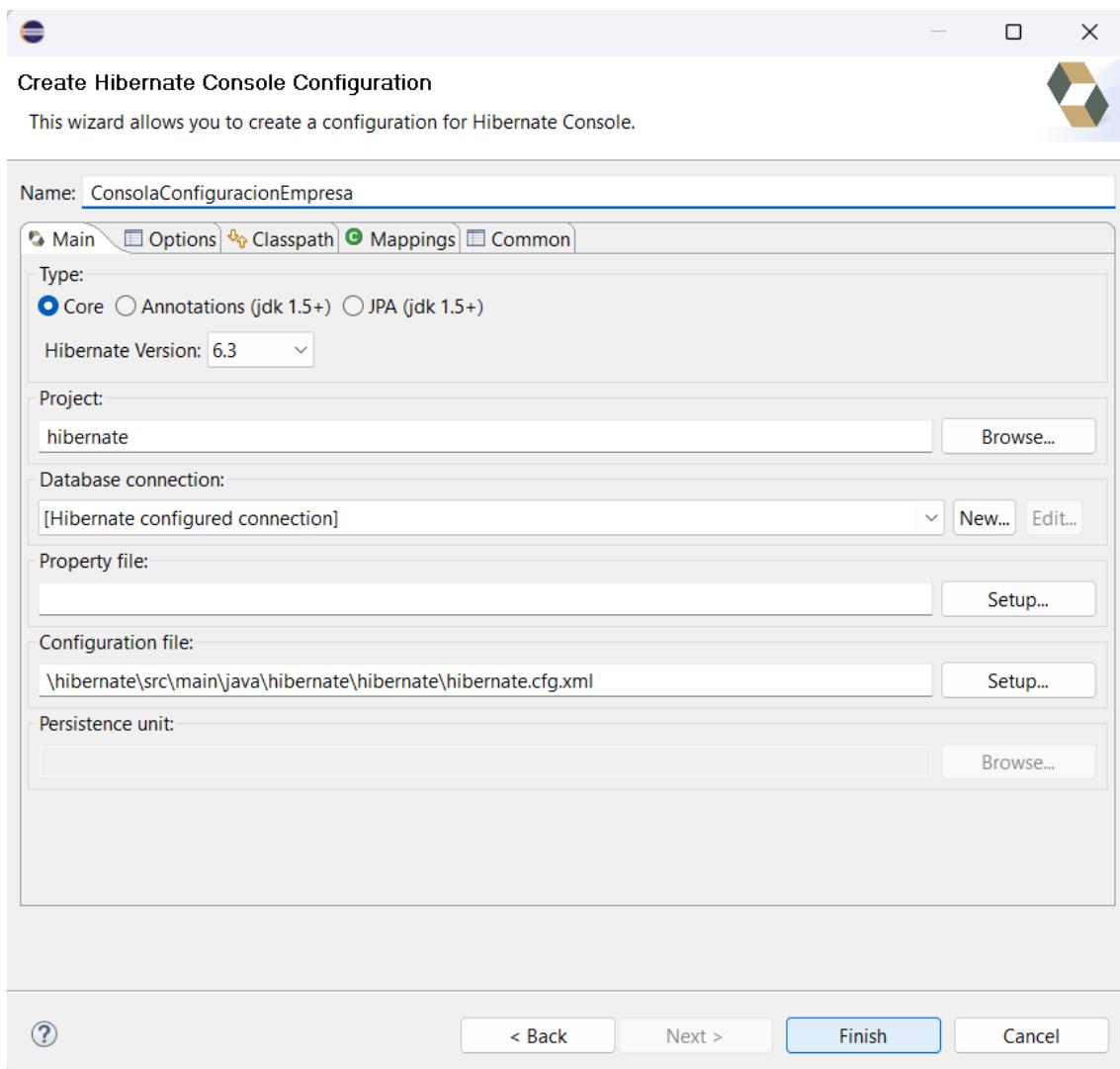
1 //Hibernate/Hibernate Configuration DTD 3.0//EN (doctype with catalog)
2 <?xml version="1.0" encoding="UTF-8"?>
3 <!DOCTYPE hibernate-configuration PUBLIC
4   "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
5   "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
6<hibernate-configuration>
7   <session-factory name="SesionEmpresa">
8     <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
9     <property name="hibernate.connection.url">jdbc:mysql://localhost/empresa</property>
10    <property name="hibernate.connection.username">root</property>
11    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
12 </session-factory>
13 </hibernate-configuration>

```

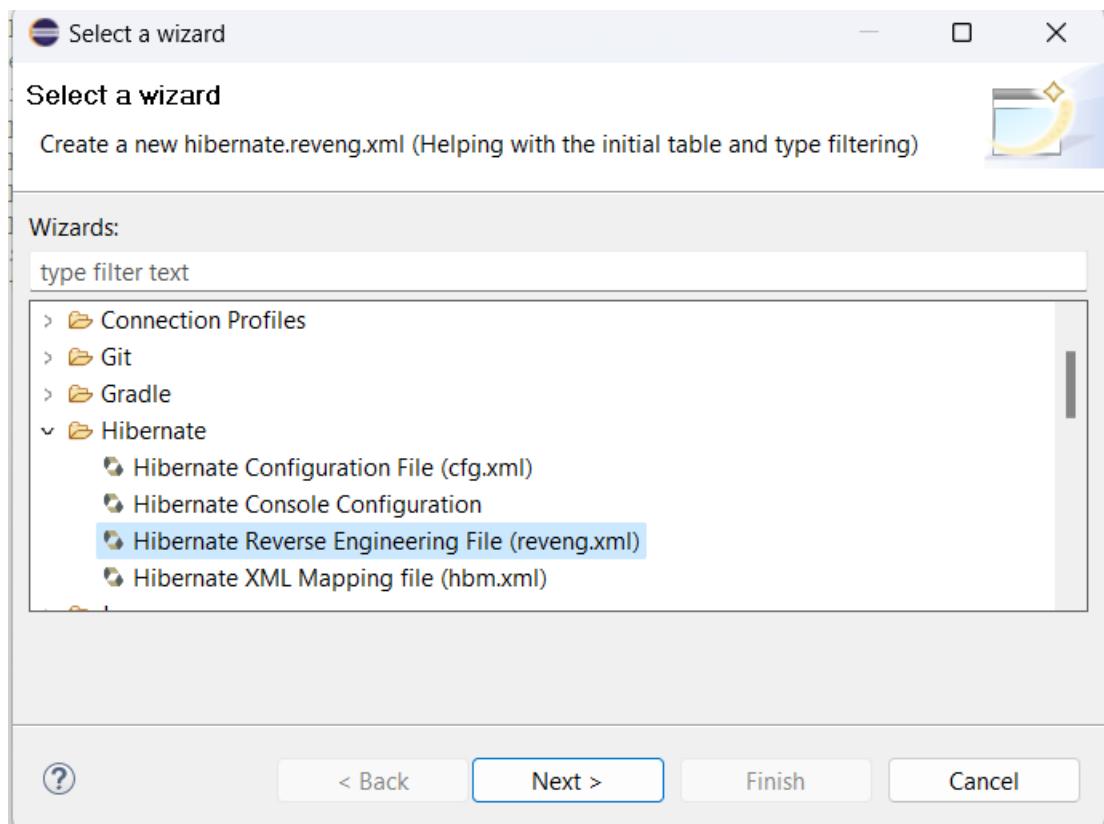
Como podemos observar ya se nos ha creado la sesión con nuestros datos de conexión.



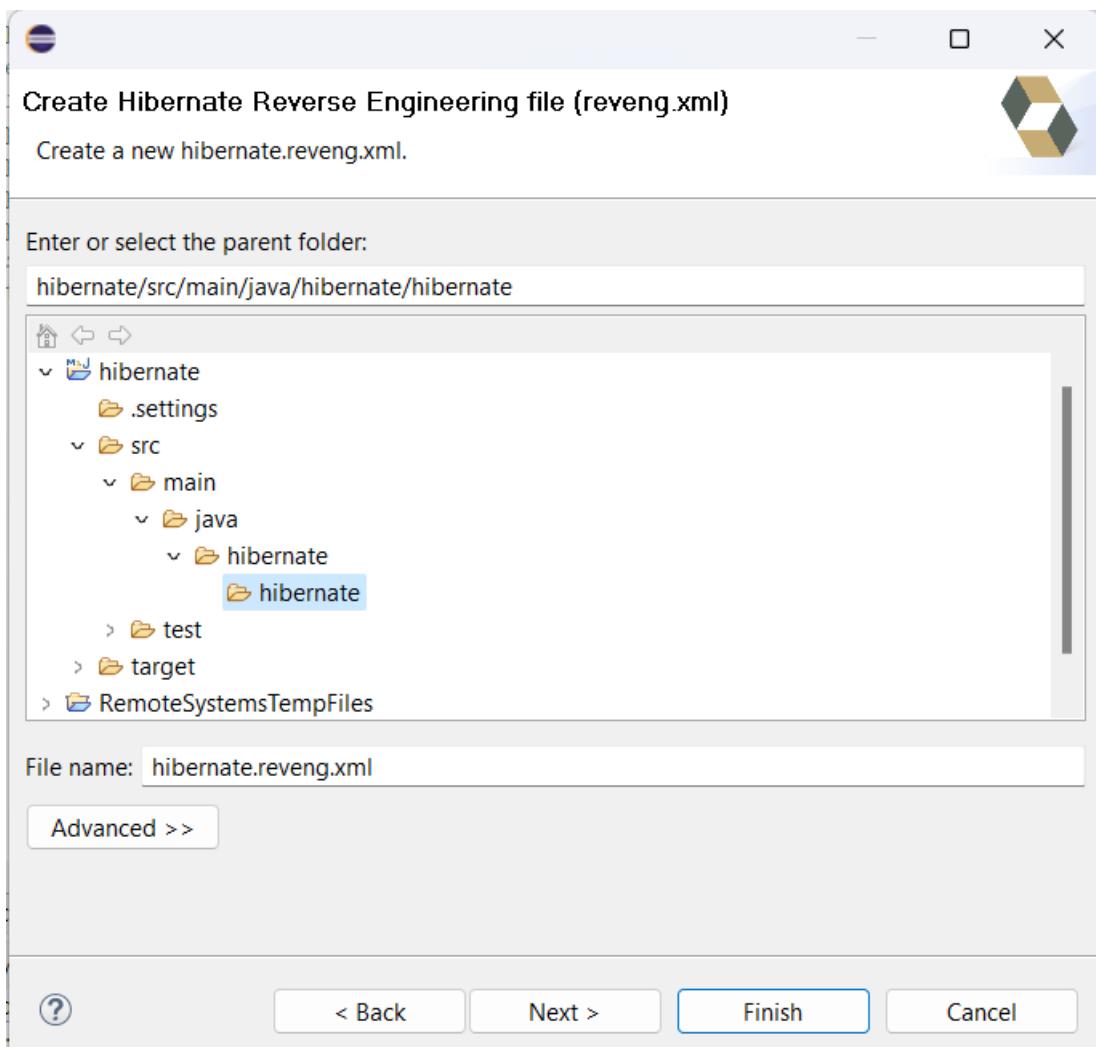
Lo siguiente que tenemos que hacer es crear otro fichero, pero en este caso será de **Hibernate Console Configuration**.



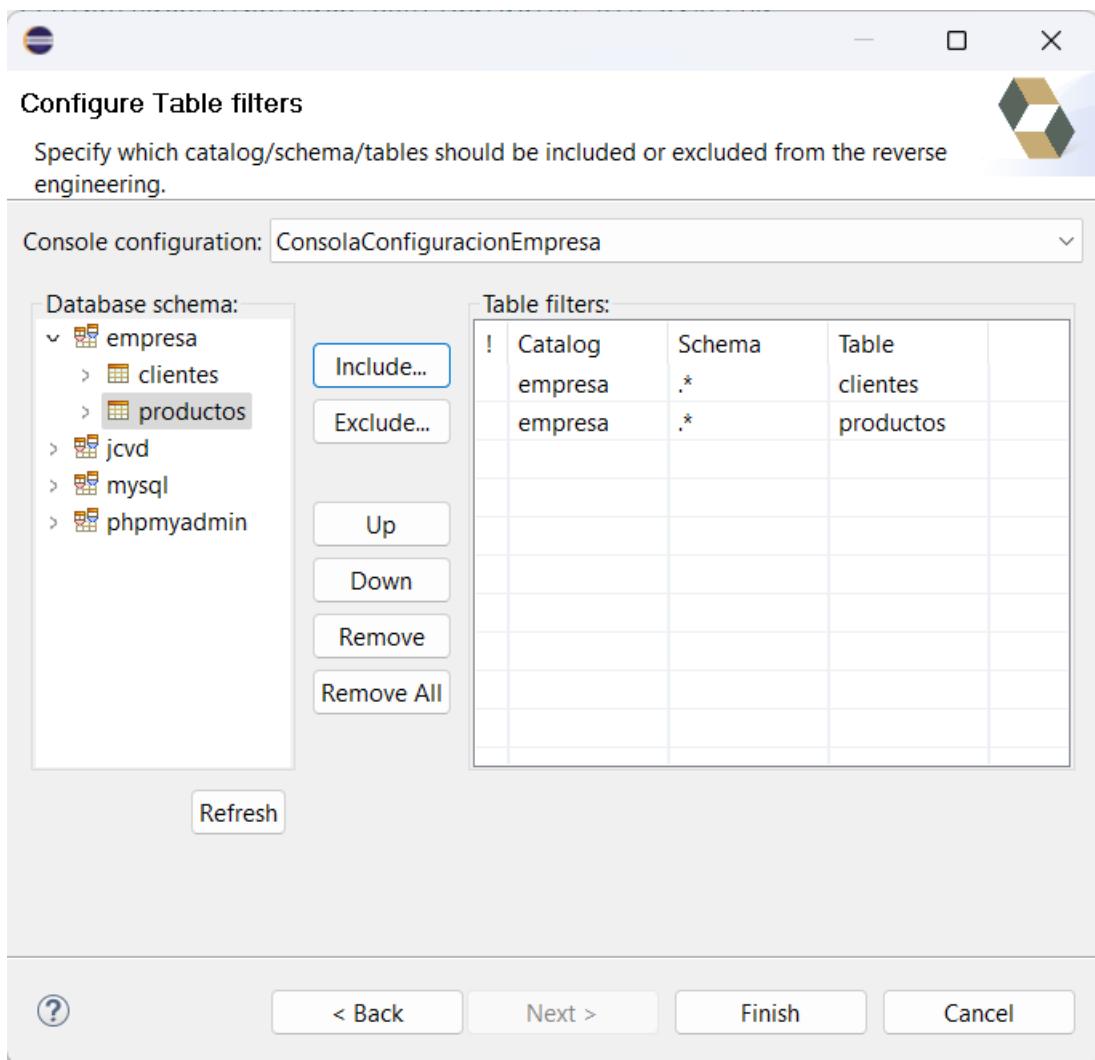
Le ponemos un nombre y en caso de que faltase algún dato lo rellenamos en este caso el archivo de configuración y el proyecto los ha puesto el directamente por el paso anterior. Le damos por último a Finish. No muestra ningún Mensaje, pero la consola se crea correctamente.



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 empresas y elegimos las dos tablas y le damos a Include para que los incluya en los filtros y por últimos le damos a Finish.

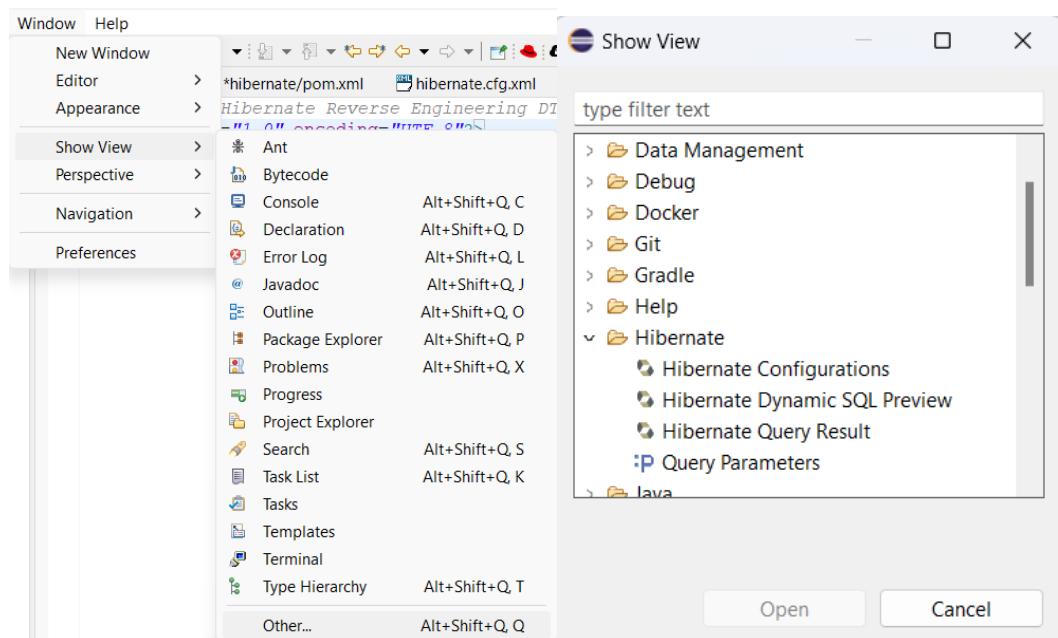
```
-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN (doctype with catalog)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN"
3
4<hibernate-reverse-engineering>
5   <table-filter match-catalog="empresa" match-name="clientes"/>
6   <table-filter match-catalog="empresa" match-name="productos"/>
7 </hibernate-reverse-engineering>
```

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

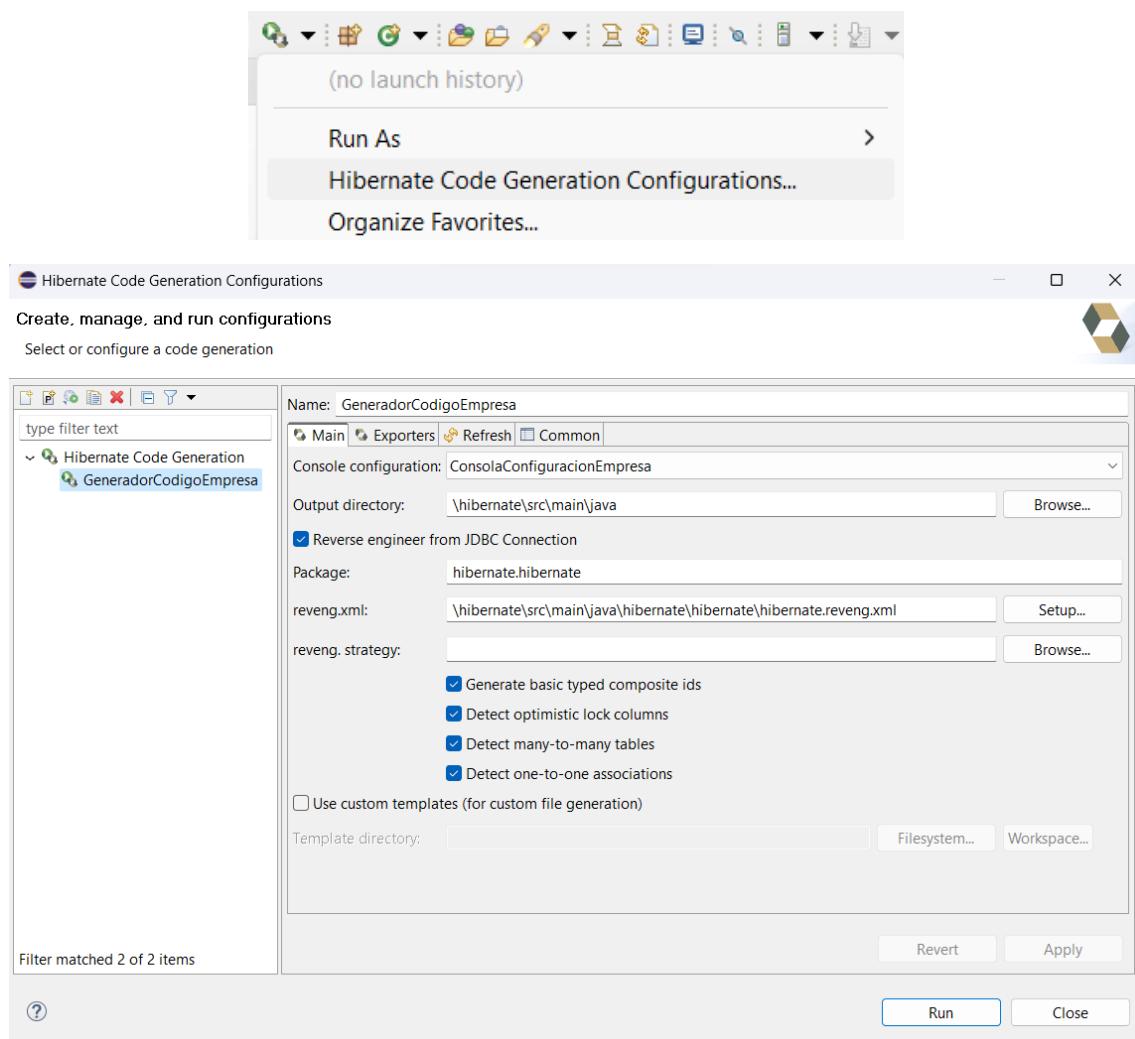
Hibernate

Eclipse

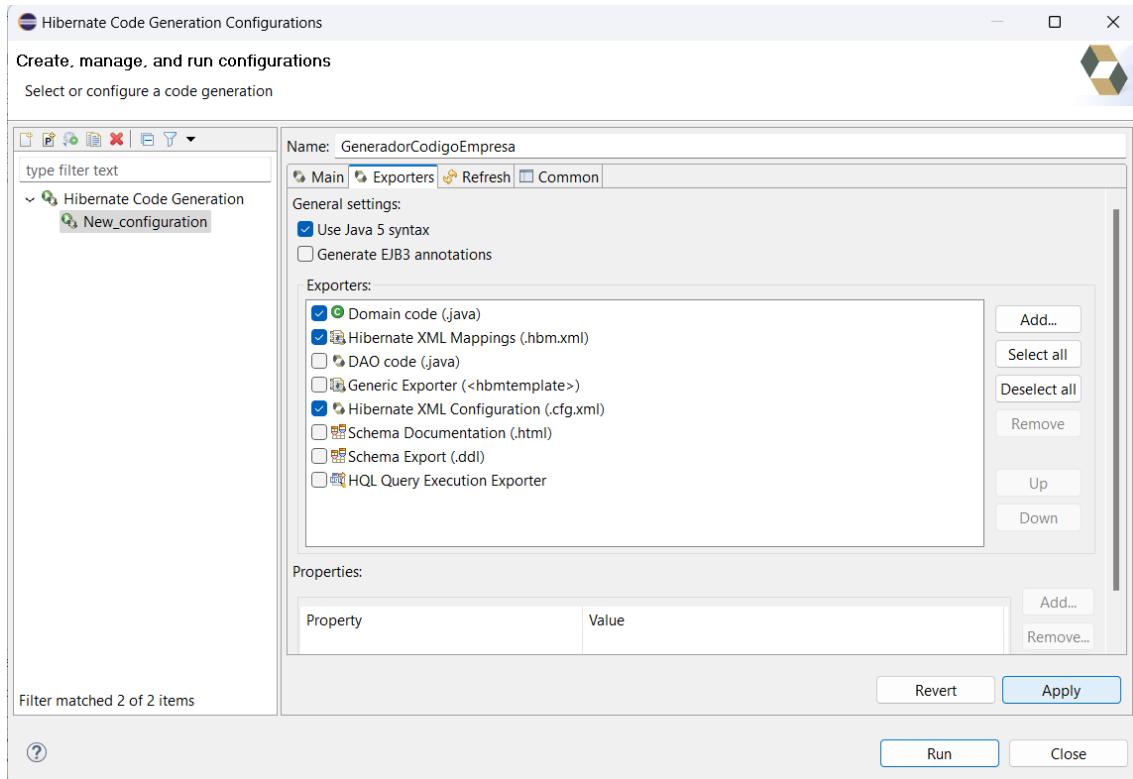
David Cristian Piroscă



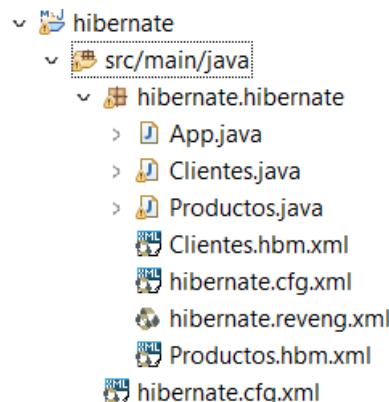
Window > Show View > Other > Hibernate > Hibernate Configurations



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.

```
1 package hibernate.hibernate;
2 // Generated 24 dic 2023, 21:01:11 by Hibernate Tools 6.3.1.Final
3
4 /**
5  * Clientes generated by hbm2java
6 */
7 public class Clientes implements java.io.Serializable {
8
9     private Integer id;
10    private String nombre;
11    private String pais;
12
13    public Clientes() {
14    }
15
16    public Clientes(String nombre, String pais) {
17        this.nombre = nombre;
18        this.pais = pais;
19    }
20
21    public Integer getId() {
22        return this.id;
23    }
24
25    public void setId(Integer id) {
26        this.id = id;
27    }
28
29    public String getNombre() {
30        return this.nombre;
31    }
32}
```

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

```

1 package hibernate.hibernate;
2 // Generated 24 dic 2023, 21:01:11 by Hibernate Tools 6.3.1.Final
3
4 /**
5  * Productos generated by hbm2java
6 */
7 public class Productos implements java.io.Serializable {
8
9     private Integer id;
10    private String nombre;
11    private double precio;
12
13    public Productos() {
14    }
15
16    public Productos(String nombre, double precio) {
17        this.nombre = nombre;
18        this.precio = precio;
19    }
20
21    public Integer getId() {
22        return this.id;
23    }
24
25    public void setId(Integer id) {
26        this.id = id;
27    }
28
29    public String getNombre() {
30        return this.nombre;
31    }

```

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

```

hibernate.cfg.xml ×
1 //Hibernate/Hibernate Configuration DTD 3.0//EN (doctype with catalog)
2 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
3 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4 <hibernate-configuration>
5     <session-factory>
6         <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
7         <property name="hibernate.connection.url">jdbc:mysql://localhost/empresa</property>
8         <property name="hibernate.connection.username">root</property>
9         <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
10        <property name="hibernate.hbm2ddl.auto">none</property>
11        <property name="hibernate.search.autoregister_listeners">true</property>
12        <property name="hibernate.validator.apply_to_ddl">false</property>
13        <mapping resource="hibernate/hibernate/Productos.hbm.xml"/>
14        <mapping resource="hibernate/hibernate/Clientes.hbm.xml"/>
15    </session-factory>
16 </hibernate-configuration>

```

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

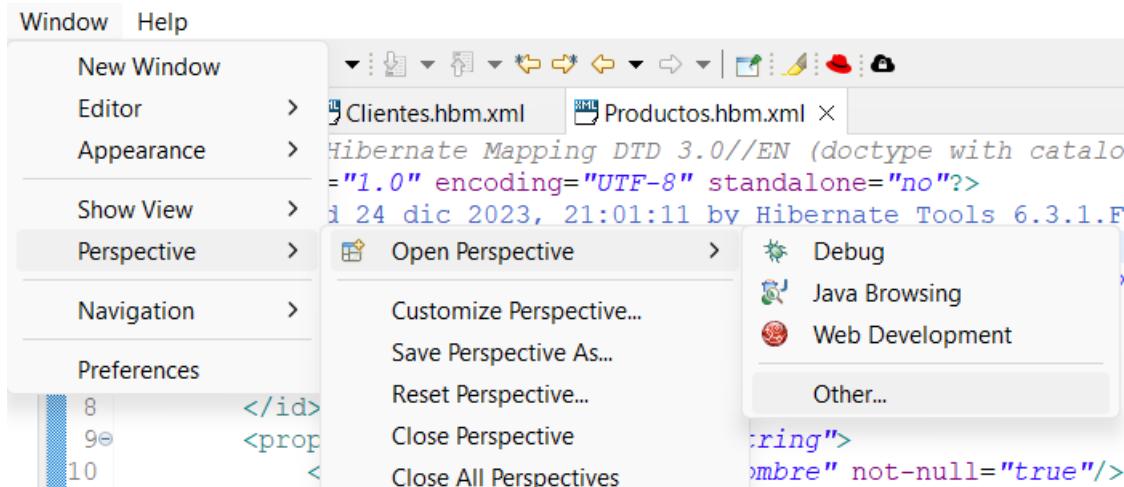
```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!-- Generated 24 dic 2023, 21:01:11 by Hibernate Tools 6.3.1.Final --><!DOCTYPE hibernate-mapping E
3<hibernate-mapping>
4<class catalog="empresa" name="hibernate.hibernate.Clientes" optimistic-lock="none" table="clie
5<id name="id" type="java.lang.Integer">
6<column name="id"/>
7<generator class="identity"/>
8</id>
9<property name="nombre" type="string">
10<column length="20" name="nombre" not-null="true"/>
11</property>
12<property name="pais" type="string">
13<column length="15" name="pais" not-null="true"/>
14</property>
15</class>
16</hibernate-mapping>

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!-- Generated 24 dic 2023, 21:01:11 by Hibernate Tools 6.3.1.Final --><!DOCTYPE hibernate-mapping E
3<hibernate-mapping>
4<class catalog="empresa" name="hibernate.hibernate.Productos" optimistic-lock="none" table="prod
5<id name="id" type="java.lang.Integer">
6<column name="id"/>
7<generator class="identity"/>
8</id>
9<property name="nombre" type="string">
10<column length="20" name="nombre" not-null="true"/>
11</property>
12<property name="precio" type="double">
13<column name="precio" not-null="true" precision="22" scale="0"/>
14</property>
15</class>
16</hibernate-mapping>

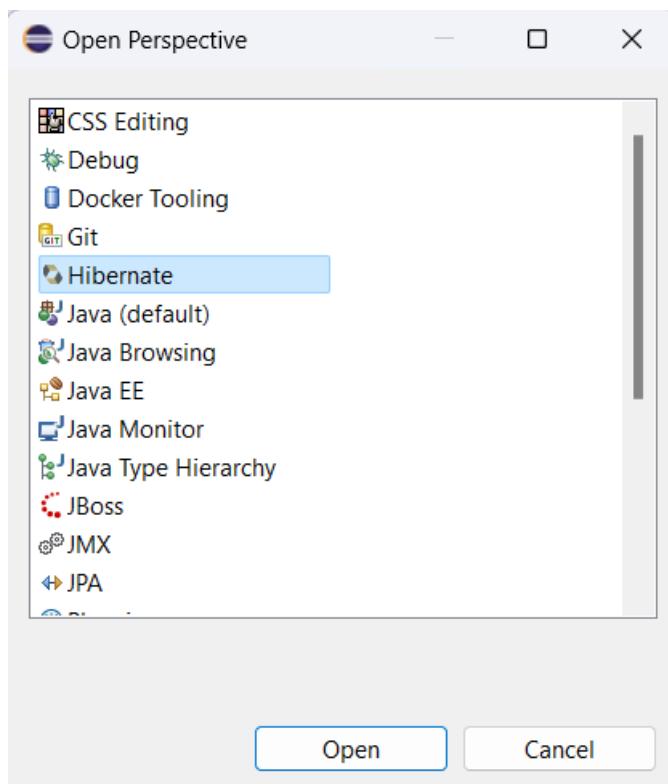
```

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

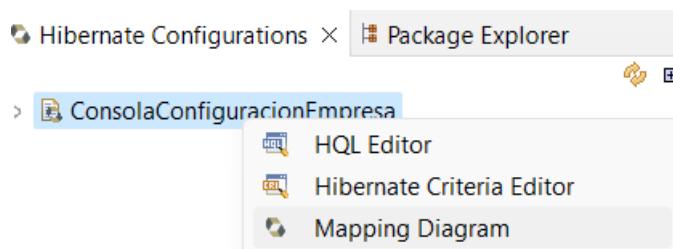


Abrimos la perspectiva de Hibernate para comprobar que la conexión con la base de datos funciona correctamente.

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 sería 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>

Hibernate ORM Hibernate Core » 6.4.1.Final

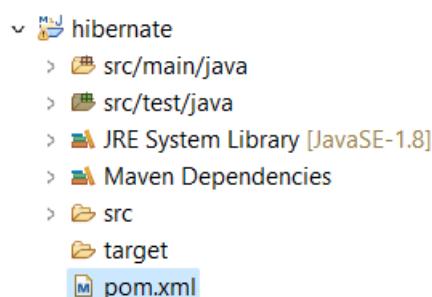
Hibernate's core ORM functionality

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

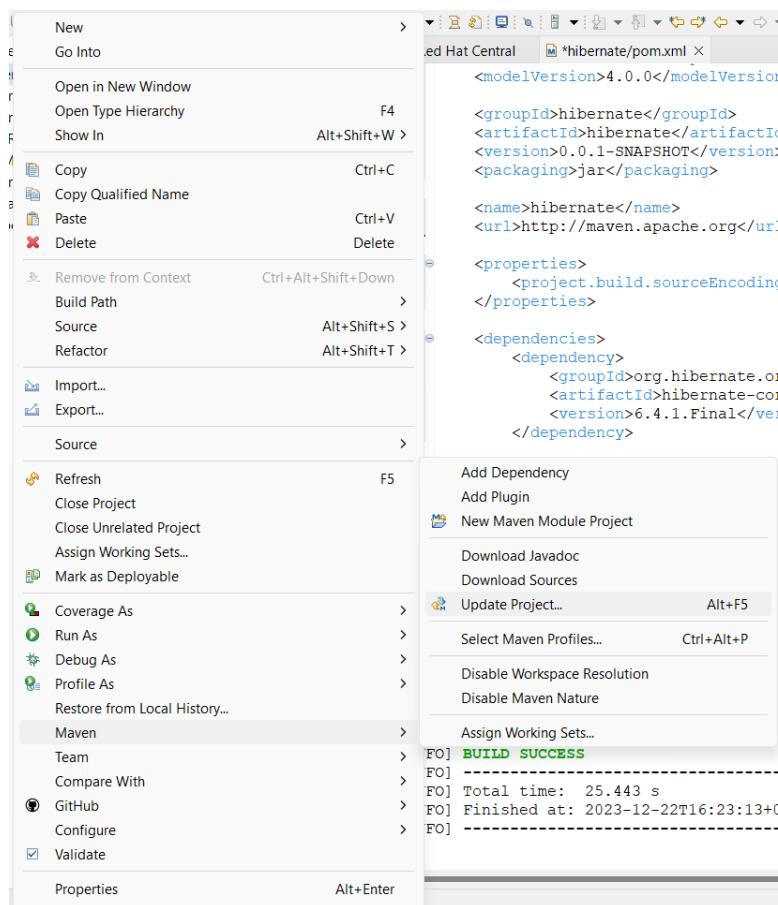
Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
<!-- 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>
```

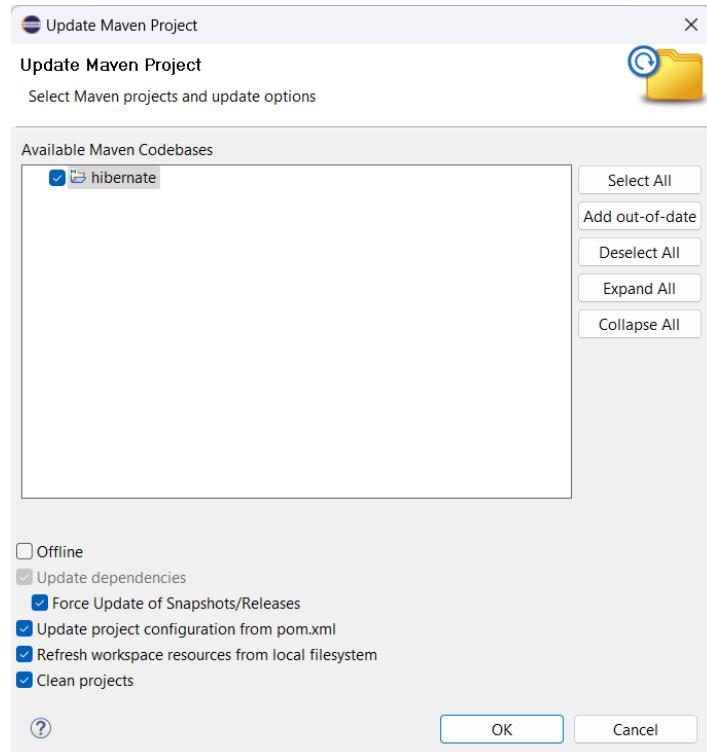
Copiamos las dependencias en el pom de nuestro proyecto Maven.



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

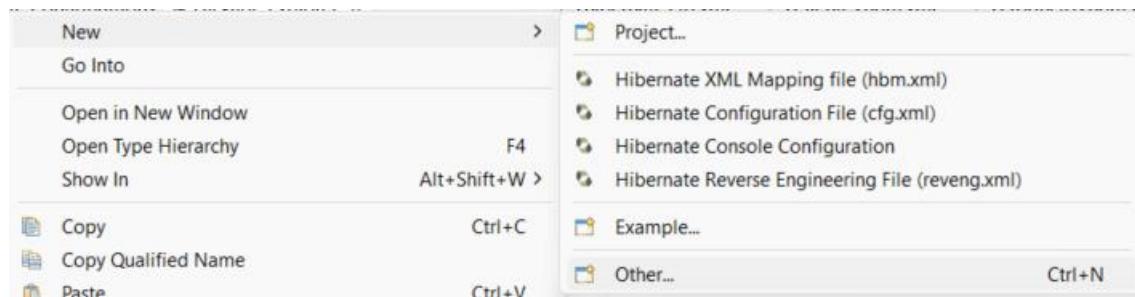


Clic derecho sobre el proyecto > Maven > Update Project.

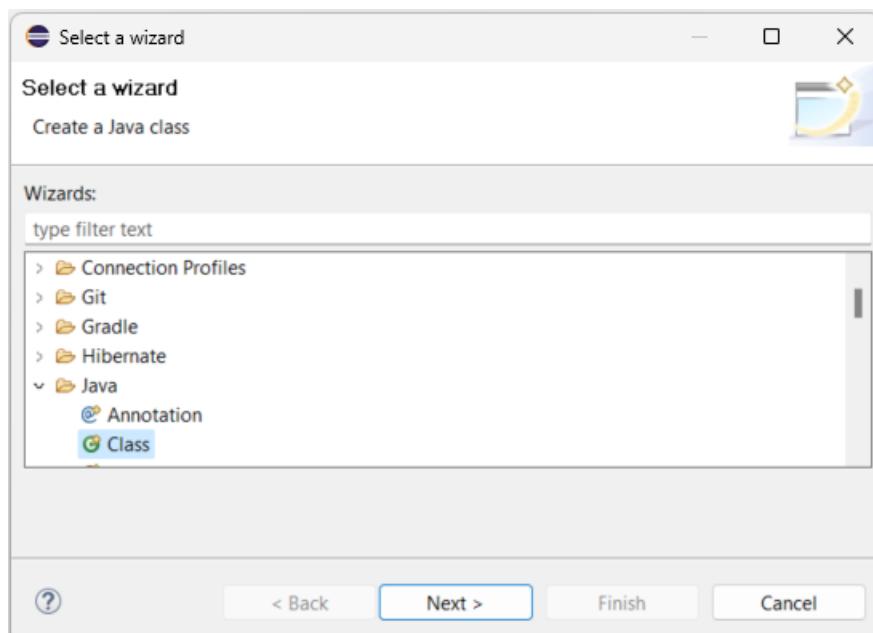


Le damos al Check Force Update of Snapshots/Releases y luego a OK. Para que se actualicen las dependencias y bajen la nueva dependencia.

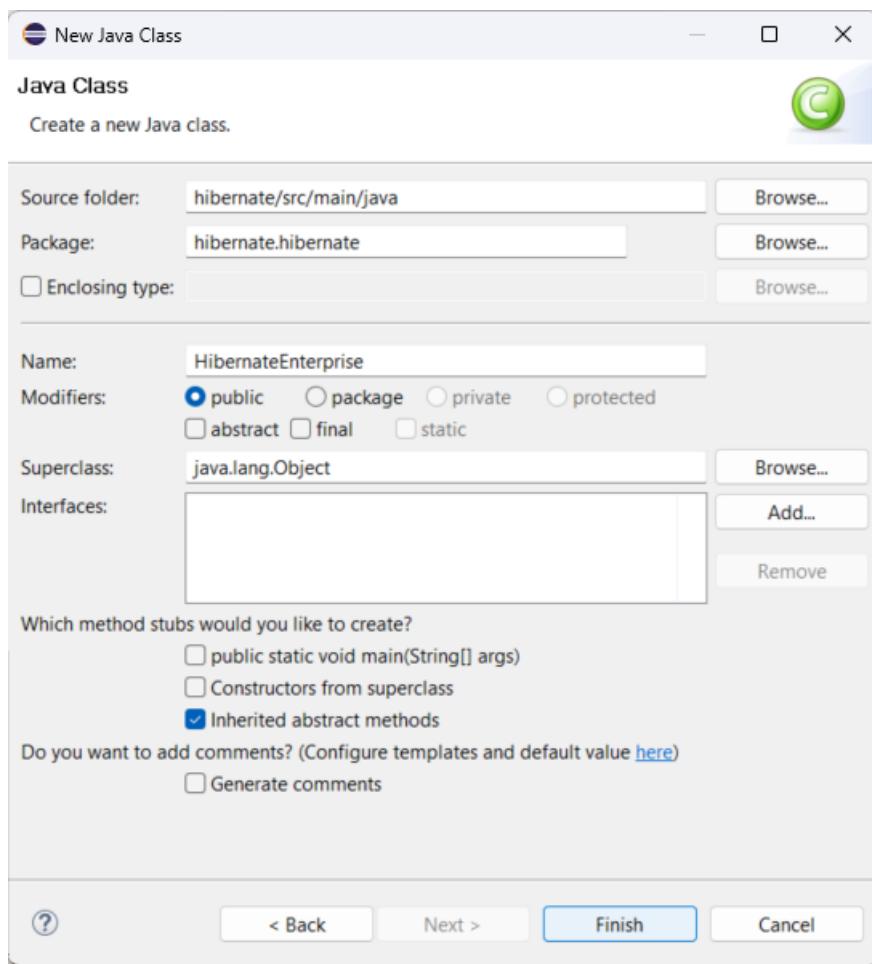
Usar Hibernate para hacer operaciones en la Base de Datos.



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 28esi be created once and used for all the
connections
    private static Productos p;

    HibernateEnterprise() {// constructor
        // sf = HibernateUtil.getSessionFactory();
        sf = new Configuration().configure().buildSessionFactory(); // also 28esió
    }

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

```

public void addProduct(String name, double 29esió) {
    Session 29esión = sf.openSession(); // 29esión es la variable que tiene el método
                                         // 
    29esi para guardar productos

    Transaction tx = null;
    // 29esión the 29esió29o with the parameters in the method
    Productos p = new Productos();
    p.setNombre(name);
    p.setPrecio(29esió);
    // 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, 29esió);
        System.out.println("=====");
        tx = 29esión.beginTransaction();
        29esión.save(p); // we INSERT p into the table PRODUCTS
        tx.commit(); // if 29esión.save doesn't 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 {
        29esión.close();
    }
}

public void showProducts() {
    Session 29esión = sf.openSession();
    Transaction tx = null;

    try {
        tx = 29esión.beginTransaction();
        List allproducts = 29esión.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 {
        29esión.close();
    }
}

```

```

public Productos findProductById(int id) {
    Session 30esióñ = 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 = 30esióñ.beginTransaction();
        p = (Productos) 30esióñ.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 {
        30esióñ.close();
    }
    return p;
}

public void deleteProductById(int id) {
    Productos p = new Productos();
    Session 30esióñ = sf.openSession();
    Transaction tx = null;
    try {
        System.out.println("=====");
        System.out.println("Buscando Producto con ID -> " + id);
        System.out.println("=====");
        tx = 30esióñ.beginTransaction();
        p = (Productos) 30esióñ.get(Productos.class, id);

        if (p != null) {
            System.out.println("=====");
            System.out.println("Borrando Producto de la Base de Datos...");
            System.out.println("=====");

            30esióñ.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("=====");
        } else {
            System.out.println("=====");
            System.out.println("No Se Encontro Ningun Producto con ID -> " + id);
            System.out.println("=====");
        }
    } catch (Exception e) {
}

```

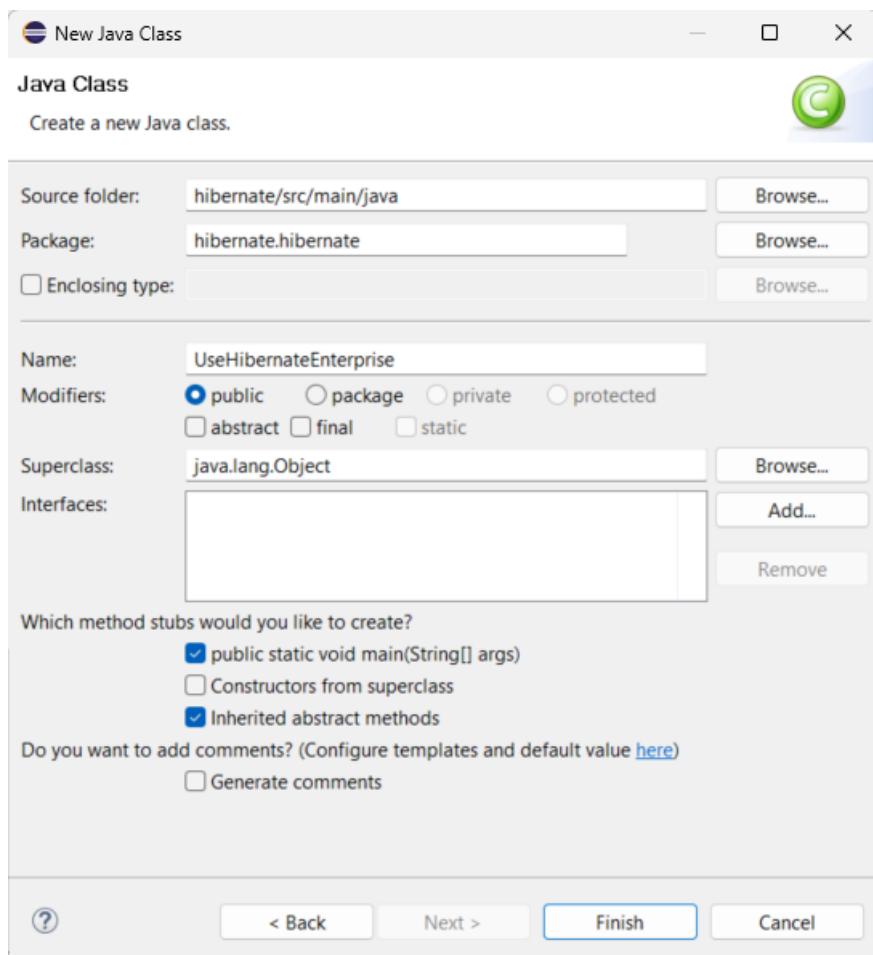
```

        if (tx != null) {
            tx.rollback();
        }
    } finally {
        sesion.close();
    }
}

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