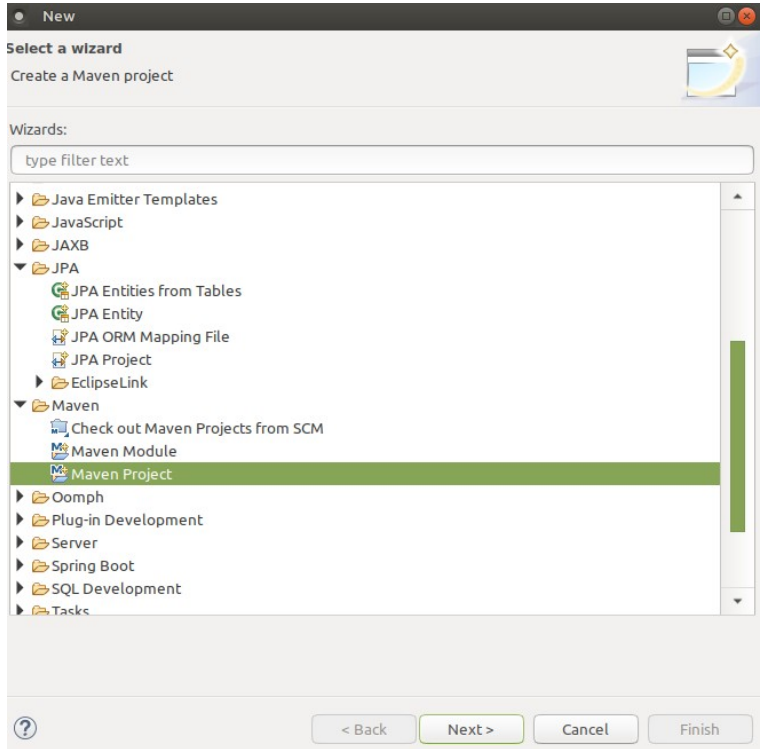
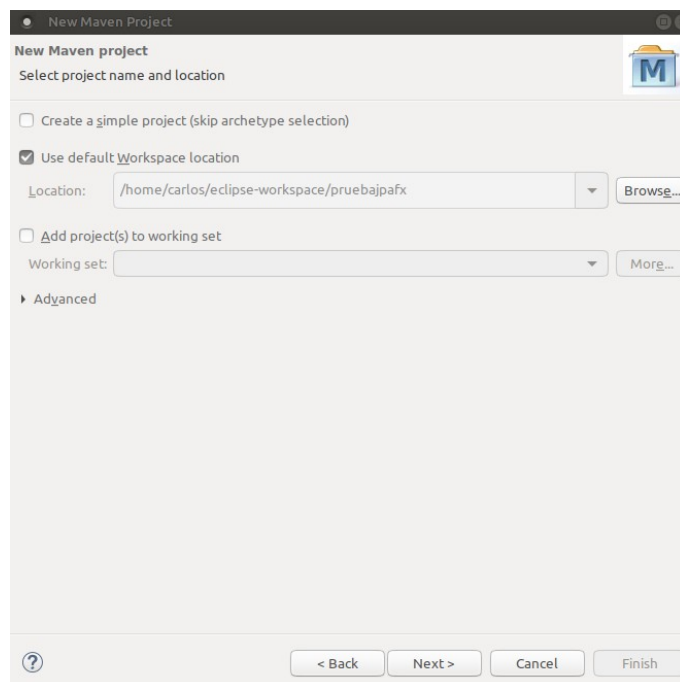


Eclipse, Maven, JavaFX, JPA

Creamos proyecto Maven (ya sabemos, new → other → maven)

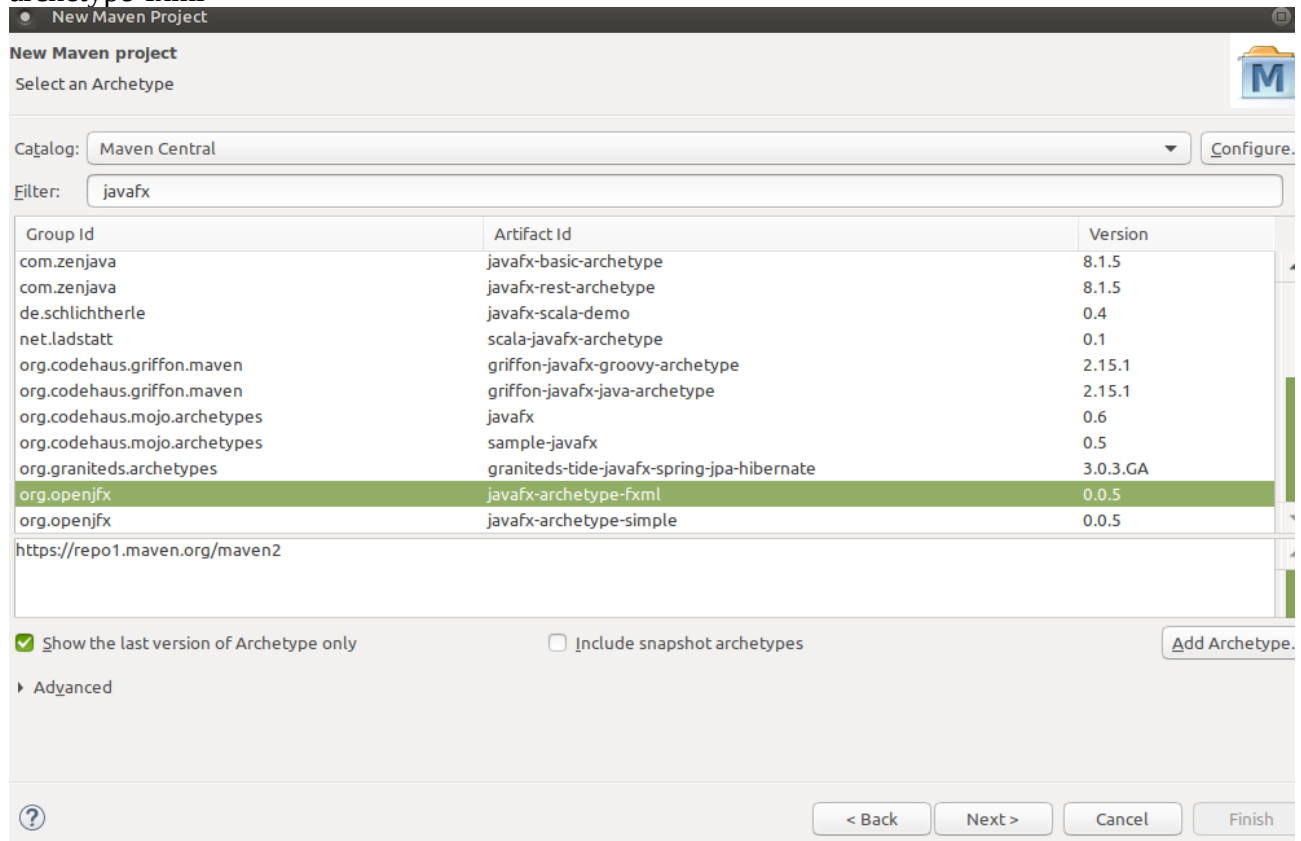


Como queremos elegir javafx no tocamos la siguiente ventana:

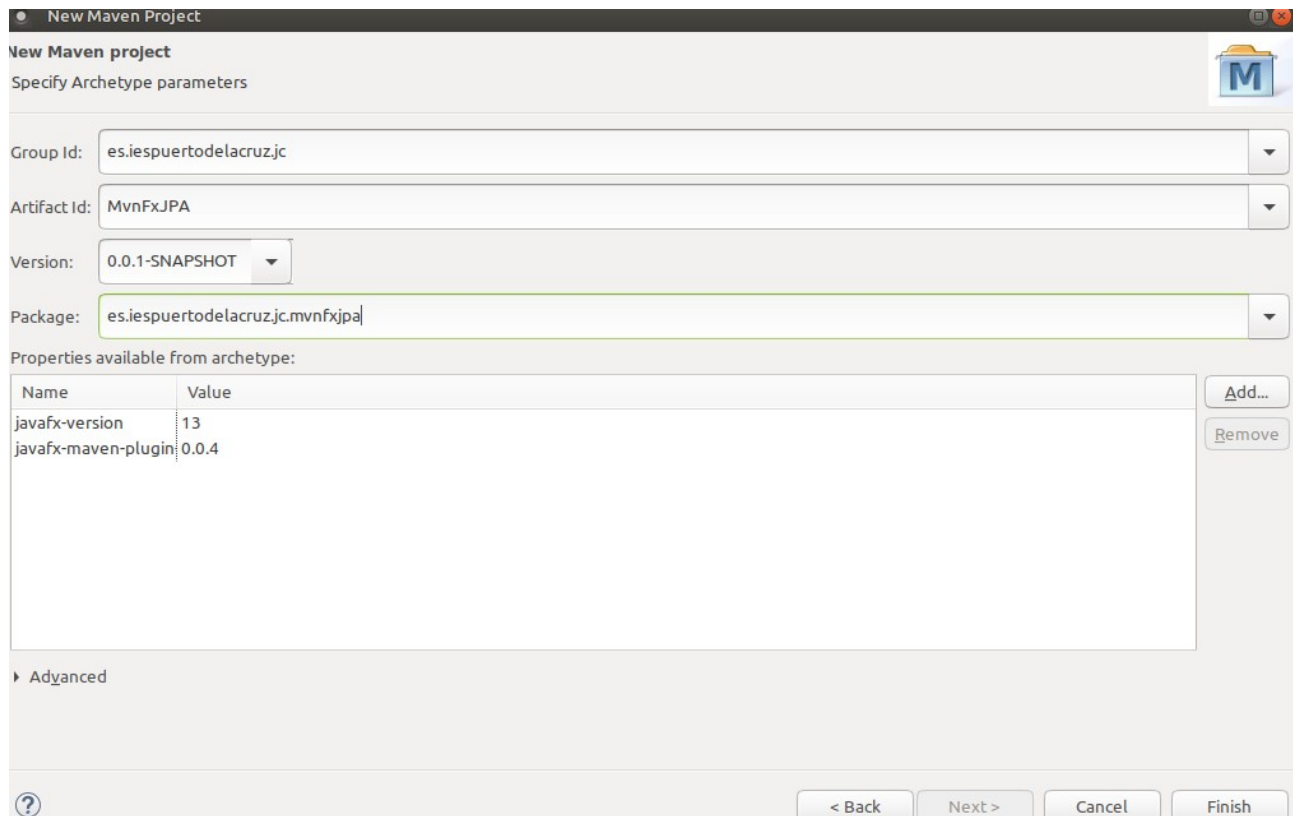


En el: Filter ponemos javafx para que nos busque y seleccionamos el penúltimo: javafx-

archetype-fxml



Lo siguiente es igual que otros proyectos maven. Definimos nuestro proyectos



Agregamos al pom.xml la parte que vamos a necesitar de bases de datos y persistencia (las dependencias únicamente, nada más)

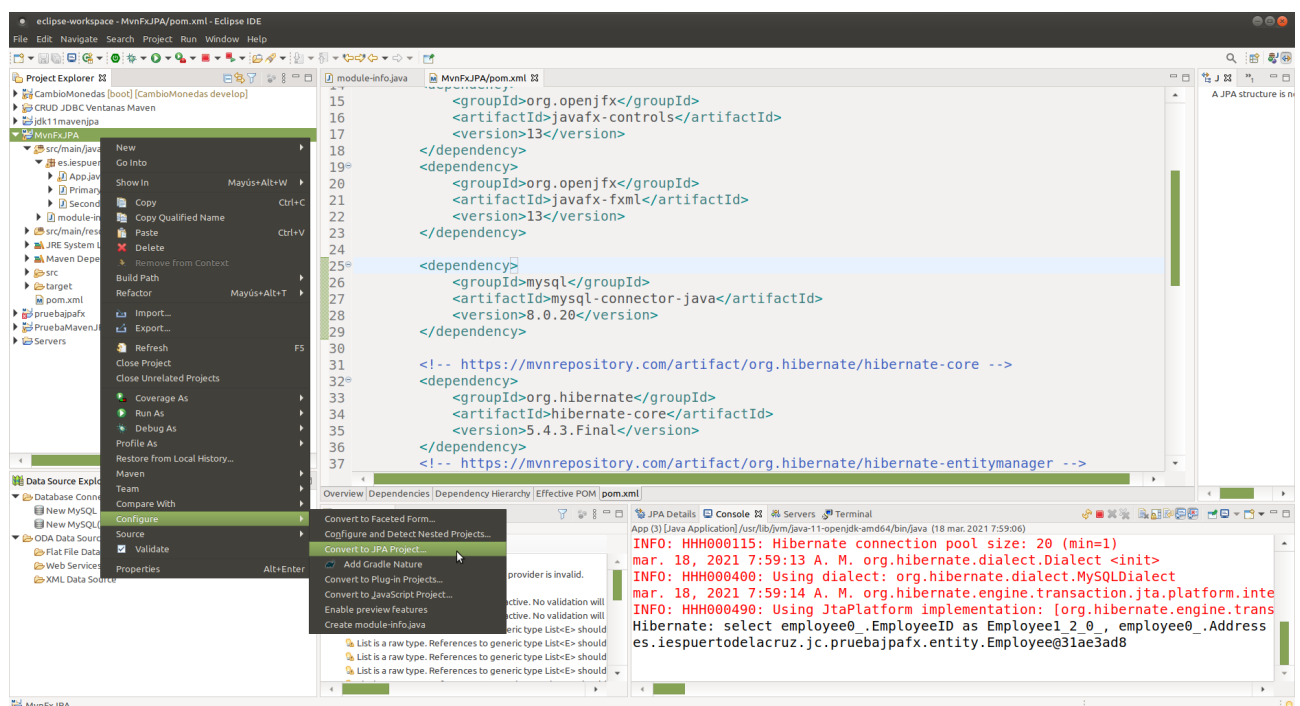
```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.20</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.4.3.Final</version>
</dependency>

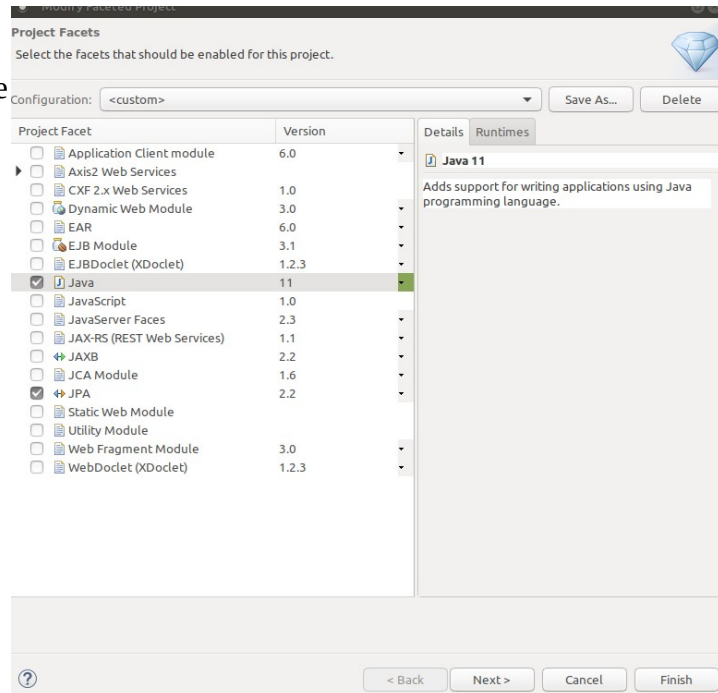
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>5.4.3.Final</version>
</dependency>

<!-- https://mvnrepository.com/artifact/javax.persistence/javax.persistence-api -->
<dependency>
  <groupId>javax.persistence</groupId>
  <artifactId>javax.persistence-api</artifactId>
  <version>2.2</version>
</dependency>
```

Convertimos el proyecto a jpa:

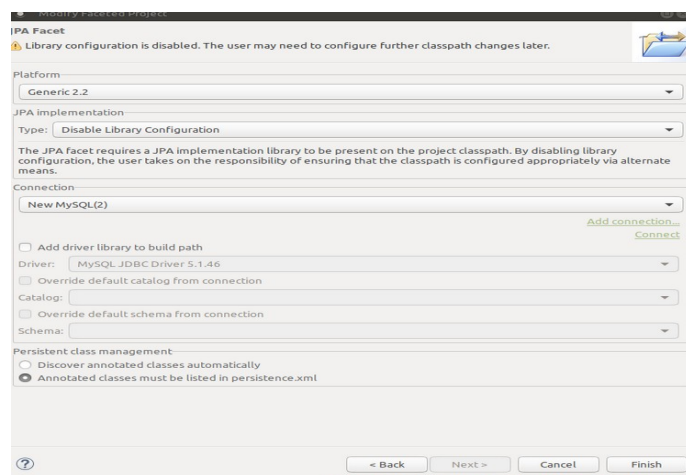


En el wizard hacemos que la versión de java sea coincidente: 11



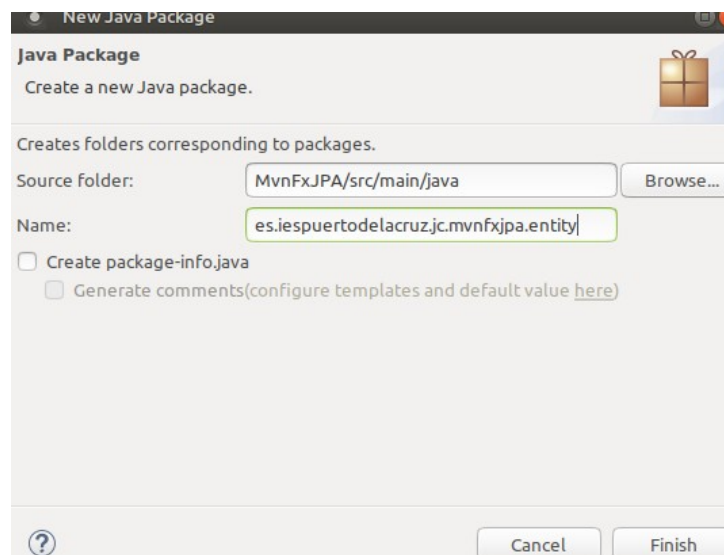
la siguiente ventana no la modificamos y pulsamos otra vez en next. Le deshabilitamos la configuración de librería y establecemos la conexión a la base de datos como hemos hecho en otros pdf

luego pulsamos finish

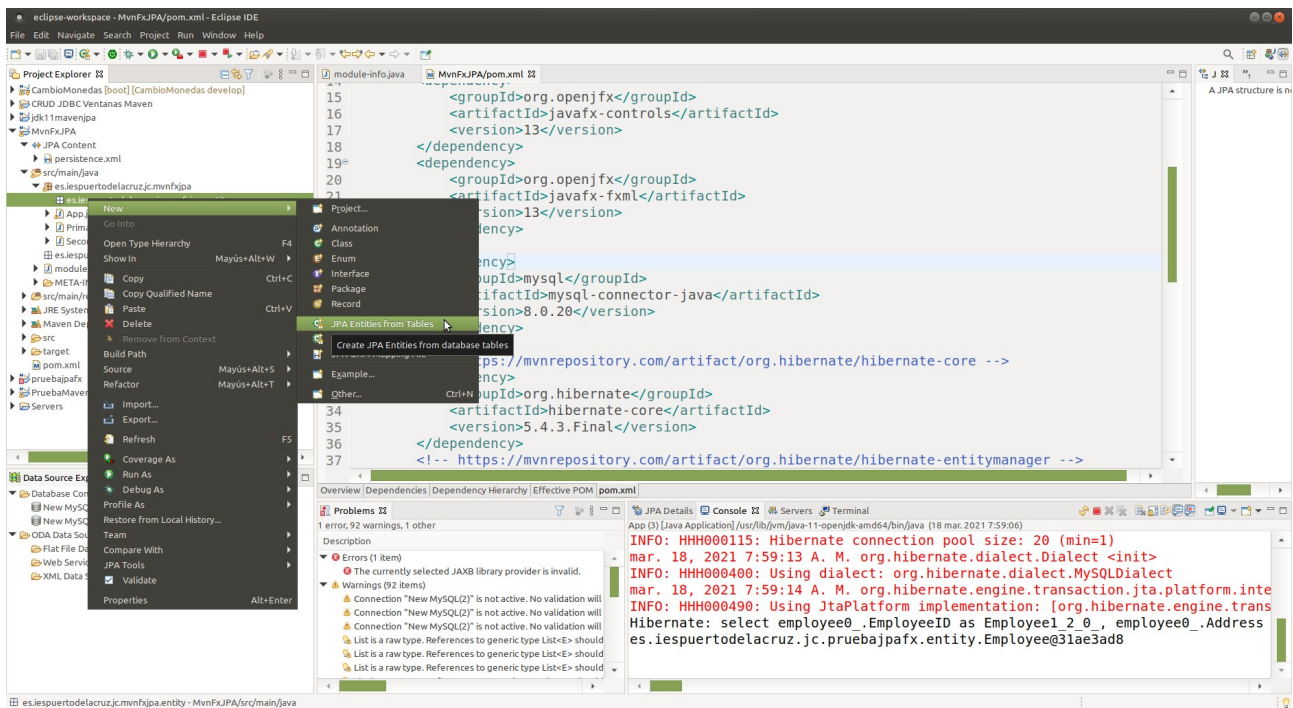


Nos habrá creado un fichero persistence.xml

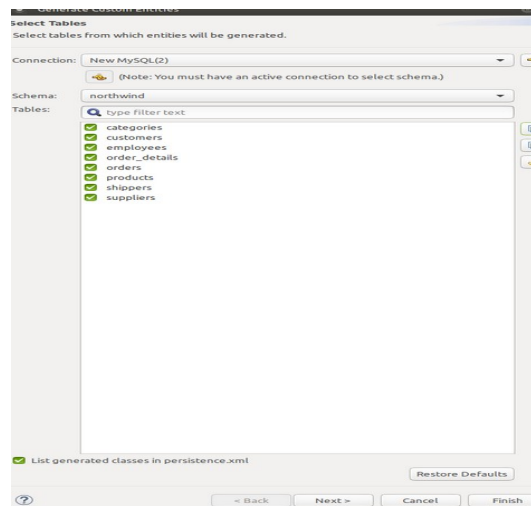
Creamos un paquete para las entities



Ahora crearemos con otro wizard las entities (esto también lo hemos hecho) con botón derecho sobre el paquete → entities from tables



En el wizard una vez puesta la conexión mysql elegimos el Schema y seleccionamos las tablas



ponemos las opciones que queremos en el wizard en la siguiente pantalla (si se quiere cascade-all)

En la que viene después están las opciones de eager, identity etc:

Customize Defaults
Optionally customize aspects of entities that will be generated by default from database tables. A Java package should be specified.

Mapping defaults

Key generator:

Sequence name:

You can use the patterns \$table and/or \$pk in the sequence name. These patterns will be replaced by the table name and the primary key column name when a table mapping is generated.

Entity access: ☒ Field ☐ Property

Associations fetch: ☐ Default ☒ Eager ☐ Lazy

Collection properties type: ☐ java.util.Set ☒ java.util.List

☒ Always generate optional JPA annotations and DDL parameters

Domain java class

Source folder:

Package:

Superclass:

Interfaces:

☒ java.io.Serializable

Cuando hayamos finalizado tenemos que corregir un par de cosas
Primero arreglar el persistence.xml
(podemos usar como base el que hemos colgado en la plataforma)

```
<persistence version="2.2"
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence

  <persistence-unit name="unidadPersistencia" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <class>es.iespuertodelacruz.jc.mvnfxjpa.entity.Category</class>
    <class>es.iespuertodelacruz.jc.mvnfxjpa.entity.Customer</class>
    <class>es.iespuertodelacruz.jc.mvnfxjpa.entity.Employee</class>
    <class>es.iespuertodelacruz.jc.mvnfxjpa.entity.OrderDetail</class>
    <class>es.iespuertodelacruz.jc.mvnfxjpa.entity.Order</class>
    <class>es.iespuertodelacruz.jc.mvnfxjpa.entity.Product</class>
    <class>es.iespuertodelacruz.jc.mvnfxjpa.entity.Shipper</class>
    <class>es.iespuertodelacruz.jc.mvnfxjpa.entity.Supplier</class>
    <properties>
      <property name="javax.persistence.jdbc.url"
        value="jdbc:mysql://localhost:3306/northwind?serverTimezone=UTC" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.driver"
        value="com.mysql.cj.jdbc.Driver" />
      <property name="javax.persistence.jdbc.password"
        value="1q2w3e4r" />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQLDialect" />
      <property name="hibernate.show_sql" value="true" />
    </properties>
  </persistence-unit>
</persistence>
```

Ahora tenemos que modificar el fichero: module-info.java

A partir de java9 empezó a trabajarse con módulos. JavaFX se convirtió en un módulo. Veamos que es lo que nos ha creado el IDE en ese fichero antes de las modificaciones:

```
module-info.java  MvnFxJPA/pom.xml  persistence.xml
1 module es.iespuertodelacruz.jc.mvnfxjpa {
2     requires javafx.controls;
3     requires javafx.fxml;
4
5     opens es.iespuertodelacruz.jc.mvnfxjpa to javafx.fxml;
6     exports es.iespuertodelacruz.jc.mvnfxjpa;
7 }
```

Vemos que dice que hace falta para la aplicación coger los módulos de javafx: controls y fxml
Observar que si estuviéramos con elementos como el htmleditor también habría que poner el modulo javafx.web

La línea que empieza por: opens
le está diciendo que paquetes de nuestra aplicación deben ser vistos por otros módulos

Así, favafx.fxml tiene que tener conocimiento de las clases que hemos puesto en nuestro paquete principal de la aplicación

Lo que ocurre es que ahora como hemos puesto cosas de persistencia hay que hacer algunas modificaciones adicionales:

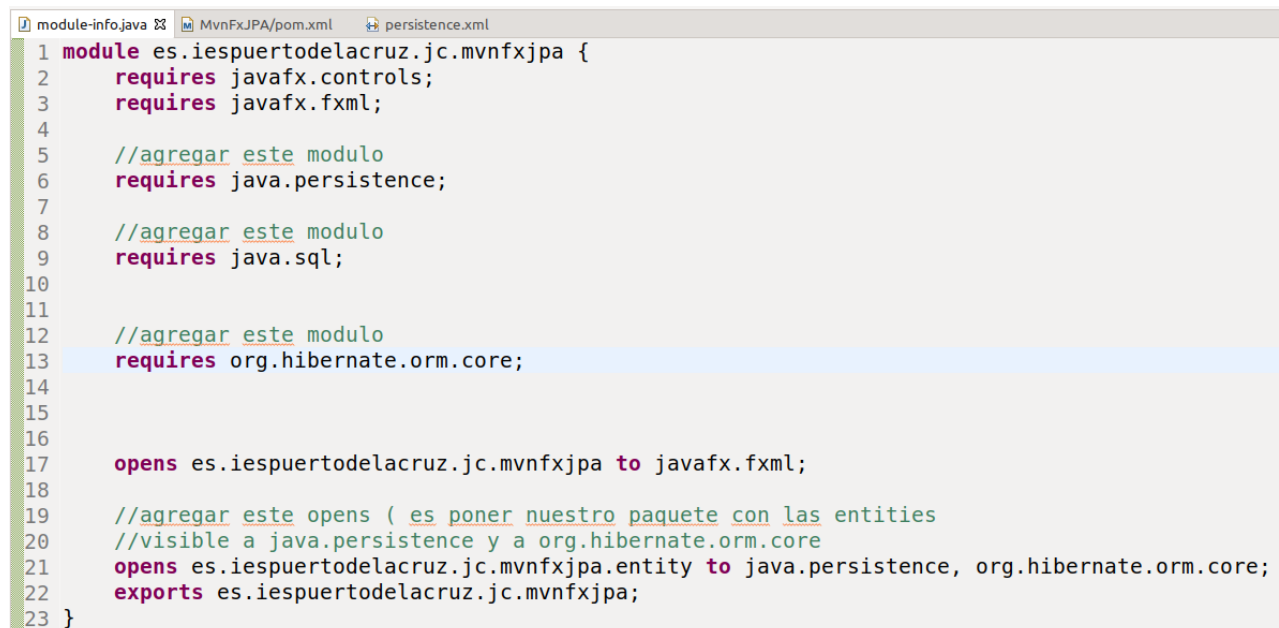
```
//agregar este modulo
requires java.persistence;

//agregar este modulo
requires java.sql;

//agregar este modulo
requires org.hibernate.orm.core;

//agregar este opens ( es poner nuestro paquete con las entities
//visible a java.persistence y a org.hibernate.orm.core
opens es.iespuertodelacruz.jc.mvnfxjpa.entity to java.persistence, org.hibernate.orm.core;
```

Nos quedará parecido a la siguiente imagen:



```
1 module es.iespuertodelacruz.jc.mvnfxjpa {
2     requires javafx.controls;
3     requires javafx.fxml;
4
5     //agregar este modulo
6     requires java.persistence;
7
8     //agregar este modulo
9     requires java.sql;
10
11
12     //agregar este modulo
13     requires org.hibernate.orm.core;
14
15
16
17     opens es.iespuertodelacruz.jc.mvnfxjpa to javafx.fxml;
18
19     //agregar este opens ( es poner nuestro paquete con las entities
20     //visible a java.persistence y a org.hibernate.orm.core
21     opens es.iespuertodelacruz.jc.mvnfxjpa.entity to java.persistence, org.hibernate.orm.core;
22     exports es.iespuertodelacruz.jc.mvnfxjpa;
23 }
```


Con lo anterior ya estaría. Como prueba vamos a agregar el siguiente código en el PrimaryController.java

```
public class PrimaryController {  
  
    @FXML  
    private void switchToSecondary() throws IOException {  
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("unidadPersistencia");  
  
        EntityManager em = emf.createEntityManager();  
        System.out.println(em.find(Employee.class, 1));  
    }  
}
```

Al ejecutar la aplicación veremos que nos muestra la dirección de memoria del empleado obtenido.