**JAVA PERSISTENCE API**

Instalación de Mysql

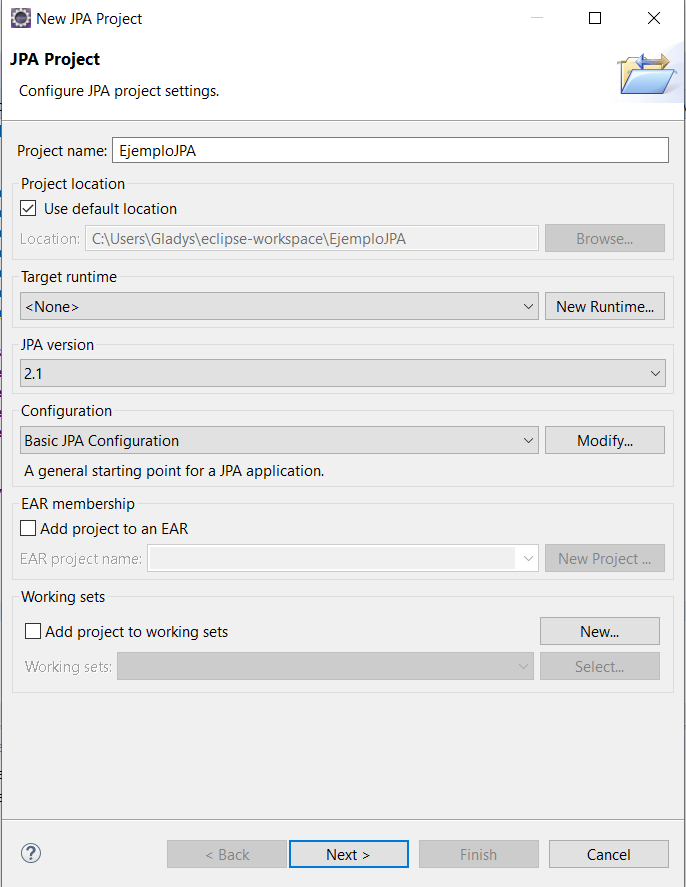
Instalar de manera local mysql.

<https://www.profesionalreview.com/2018/12/13/mysql-windows-10/>

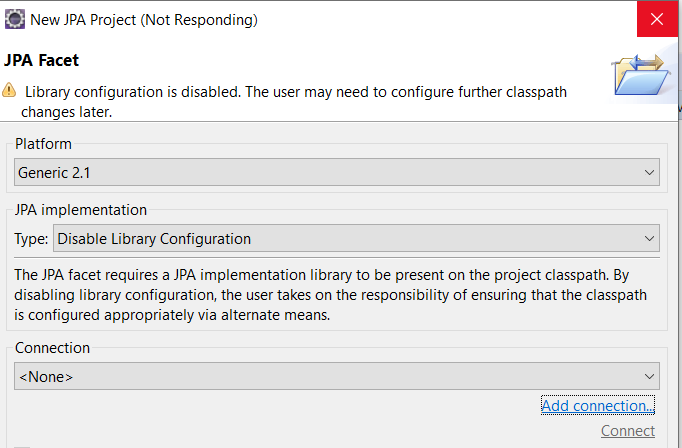
He instalado la version 8 de mysql

Si no pudo instalar el servidor mysql podrá conectarse a otra máquina que actúe como servidor.

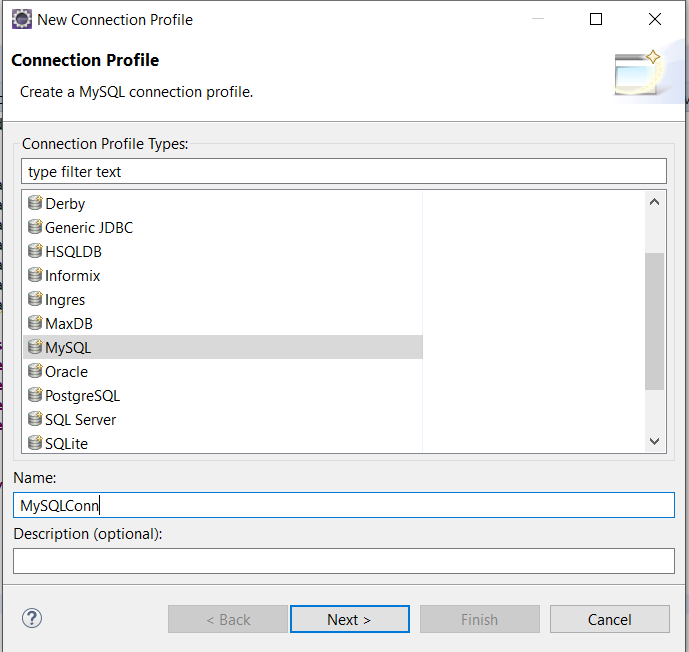
**Crear JPA Project**



En la sigueinte ventana Next

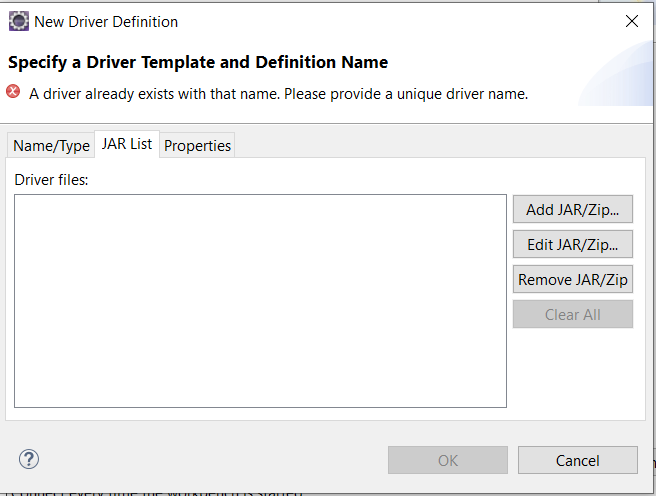


Al dar click en Add connection, se muestra una nueva ventana. Aquí se seleccionará el motor de bases de datos, para este ejemplo MySQL

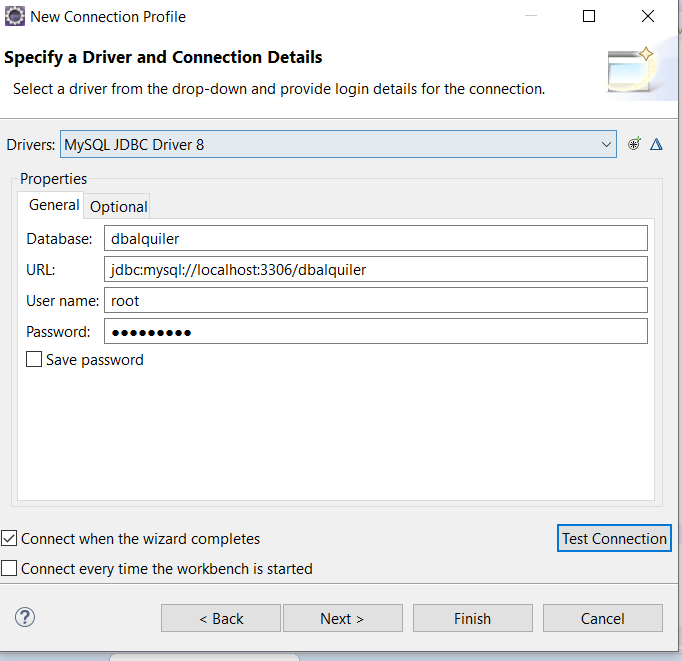


Se da click en Next

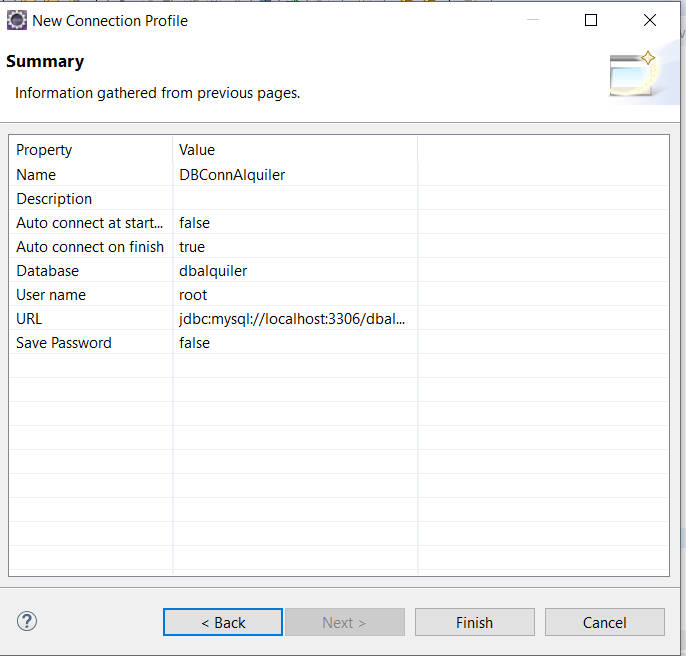
Nos pedirá que indiquemos el driver. Si no aparece algún driver en el listado hay que dar click en el más. En la nueva ventana, dar click en JAR List para poder agregar el jar con el conector mysql



Al dar click en OK se muestra el formulario para ingresar los datos de la conexión



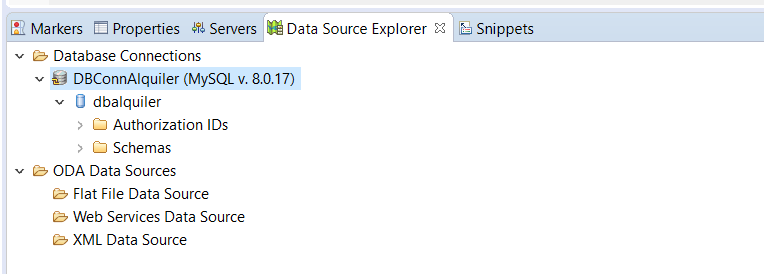
Luego de ingresar los datos, dar click en Test Connection para comprobar la conexión. Click en Next

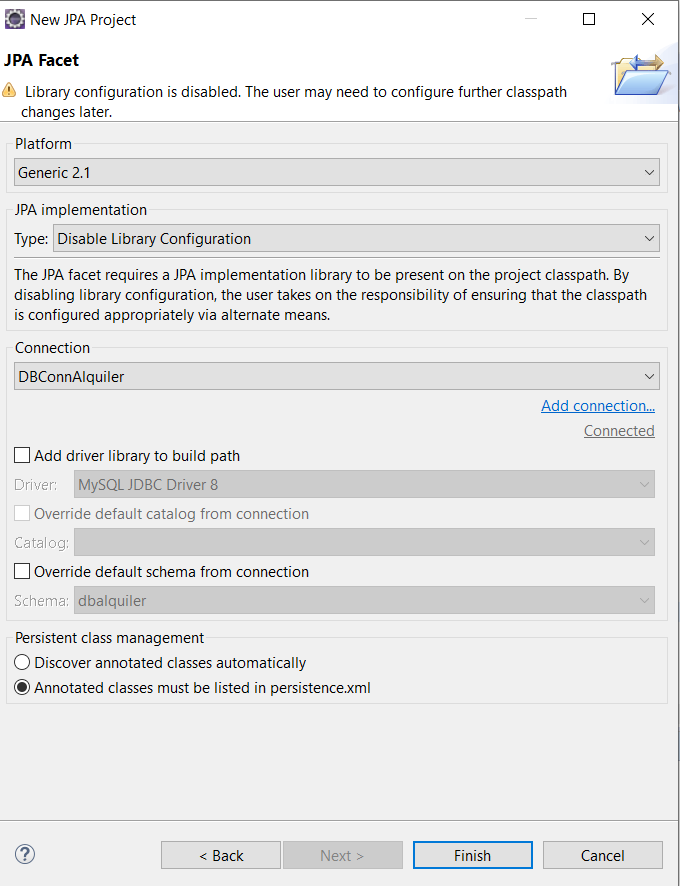


Dar click en Finish

La conexion creada ahora está disponible para ser seleccionada en la creación del proyecto JPA

La conexión a la base se puede crear previamente a través del Data Source Explorer





Luego dar click en Finalizar

Si hay un cuadro de diálogo respecto al cambio de la perspectiva dar clik en Aceptar

En la carpeta del proyecto creamos un nuevo directorio lib donde copiaremos los jar de mysql y openjpa

Descargar los jar de:

<https://github.com/gladyscarrillo/JAVAEE/tree/master/lib>

Refrescamos el proyecto y agregamos las librerias al Build Path

**archivo persistence.xml**

Agregar las propiedas que indican los datos de la conexion

<properties>

<property name="openjpa.RuntimeUnenhancedClasses" value="supported"/>

<property name="openjpa.ConnectionURL" value="jdbc:mysql://localhost:3306/dbalquiler"/>

<property name="openjpa.ConnectionDriverName" value="com.mysql.jdbc.Driver"/>

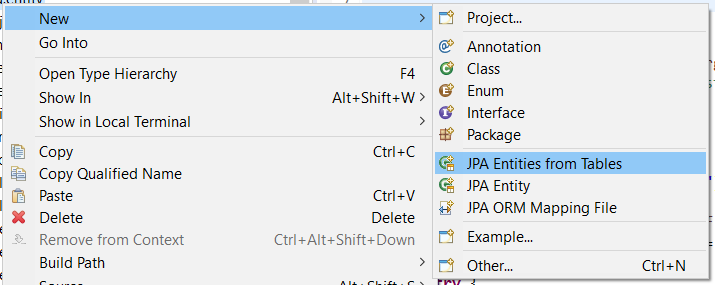
<property name="openjpa.ConnectionUserName" value="root"/>

<property name="openjpa.ConnectionPassword" value="gladys123"/>

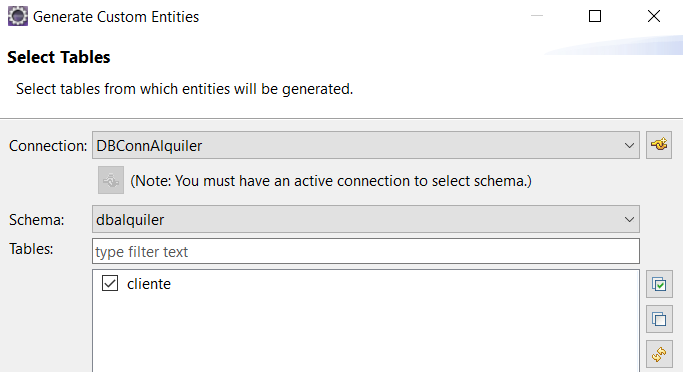
<property name="openjpa.Log" value="DefaultLevel=WARN, Tool=INFO"/>

</properties>

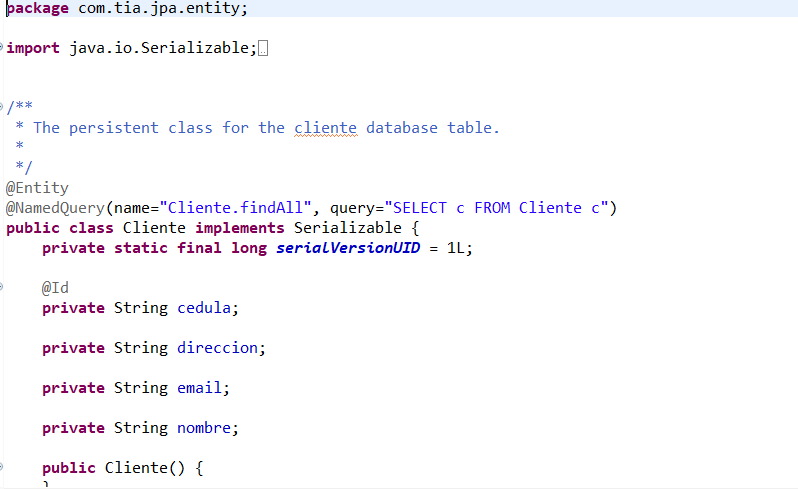
En el proyecto creamos un nuevo paquete **com.tia.jpa.entity**. En el paquete creamos un nuevo archivo para realizar nuestra entidad. Seleccionar JPA Entities from Tables



Se selecciona la conexion, se listan todas las tablas, seleccionamos la tabla cliente



Al dar clicl en Finish tenemos el archivo java con las respectivas anotaciones



En esta clase agregaremos un nuevo constructor con todos los atributos

Creamos una clase para probar. TestCliente.

En esta clase creamos un metodo para probar el insert

public static void insertar() {

System.out.println("A insertar");

EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPAEjemploTest");

EntityManager em = emf.createEntityManager();

EntityTransaction et = em.getTransaction();

try {

//cambiar los datos por los propios

Cliente cliente = new Cliente("123456787", "Isabel Rodriguez", "Luque y Chimborazo",

"isabel.rodriguez@tia.com.ec");

et.begin();

em.persist(cliente);

et.commit();

} catch (Exception e) {

System.out.println("Error:" + e.getMessage());

} finally {

em.close();

}

}

Al ejecutar, confirmamos que la tabla tenga el nuevo registro

Para editar y elimnar se hace uso del método find para recuperar por la clave primaria

Cliente cliente = em.find(Cliente.class, "123456787");

Podemos cambiar cualquier atributo y luego realizar la actualizacion en la base de datos

et.begin();

cliente.setDireccion("Chimborazo y Luque");

em.merge(cliente);

et.commit();

Para eliminar se puede utilizar el metodo remove

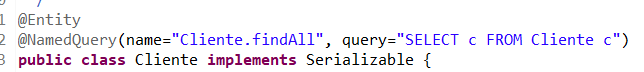
et.begin();

em.remove(cliente);

et.commit();

**CONSULTA DE DATOS**

Podemos definir consultas a realizar sobre la tabla, agregandolas con un NamedQuery, por defecto se crea findAll



Utilizaremos este query para consultar los datos presentes en la tabla

public static void consultar() {

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

EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPAEjemploTest");

EntityManager em = emf.createEntityManager();

EntityTransaction et = em.getTransaction();

try {

et.begin();

TypedQuery<Cliente> query = em.createNamedQuery("Cliente.findAll",Cliente.class);

List<Cliente> lista = query.getResultList();

et.commit();

for(Cliente c:lista) {

System.out.println(c);

}

} catch (Exception e) {

System.out.println("Error:" + e.getMessage());

} finally {

em.close();

}

}

**Agregando querys**

Agregaremos una nueva consulta para recuperar los clientes que tengan algún texto dado en su dirección. Los queries quedarian asi:

@NamedQueries({

@NamedQuery(name="Cliente.findByAddress", query="SELECT c FROM Cliente c WHERE c.direccion like :param "),

@NamedQuery(name="Cliente.findAll", query="SELECT c FROM Cliente c")

})

Para usar este query, el parametro se envia usando:

query.setParameter("param", "%Luque%");

**Referencias**

Agregar conexion base de datos en eclipse

<https://www.admfactory.com/configure-database-connection-in-eclipse/>

Estrategias para modelar herencia

<https://thoughts-on-java.org/complete-guide-inheritance-strategies-jpa-hibernate/>

<https://thoughts-on-java.org/many-relationships-additional-properties/>

<https://thoughts-on-java.org/complete-guide-inheritance-strategies-jpa-hibernate/>

<http://www.javawebtutor.com/articles/ejb/ejb3_and_jpa_example_in_eclipse.php>