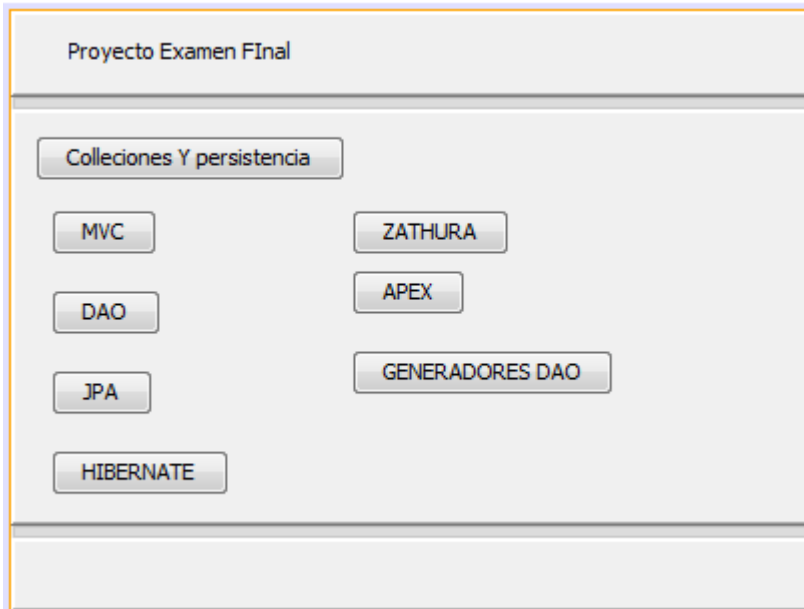


TUTORIAL JPA

RECORDEMOS EL OBJETIVO DEL EXAMEN FINAL

HACER VARIAS APLICACIONES O UNA APLICACION CON VARIOS MODULOS
Y CADA MODULO QUE GESTIONE LOS EMPLEADOS DE UN DEPARTAMENTO



La primera Opcion Colecciones y persistencia no usara bases de datos pero si persistencia de Objetos

JPA usara el patrón de diseño Modelo Vista Control

DAO usara el patrón de diseño Data Access Objects y asi sucesivamente APEX Y ZATHURA Son soluciones web.

En este Capitulo se dará un tutorial para Crear la aplicación

con bases de datos y JPA

Objetivo:

Crear un aplicación con las funcionalidades básicas para el el CRUD de las tablas Departamento y Empleado. Usando la Persistencia JPA de JAVA (java Persistence Api)

El Scrip SQL para crear la base de datos Crea.SQL se anexa

PRIMERA PARTE

En la primera parte se hará el CRUD para la tabla departamentos

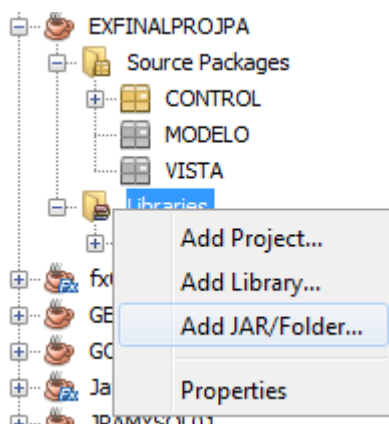
INSTRUCCIONES

PASO 1 : Cree un proyecto EXFINALPROJPA

y debe tener los tres packages MODELO VISTA CONTROL

Incluya (copia) en el package Control las librerías de Oracle en mi caso ojdbc6.jar

Y adicione las librerías al proyecto

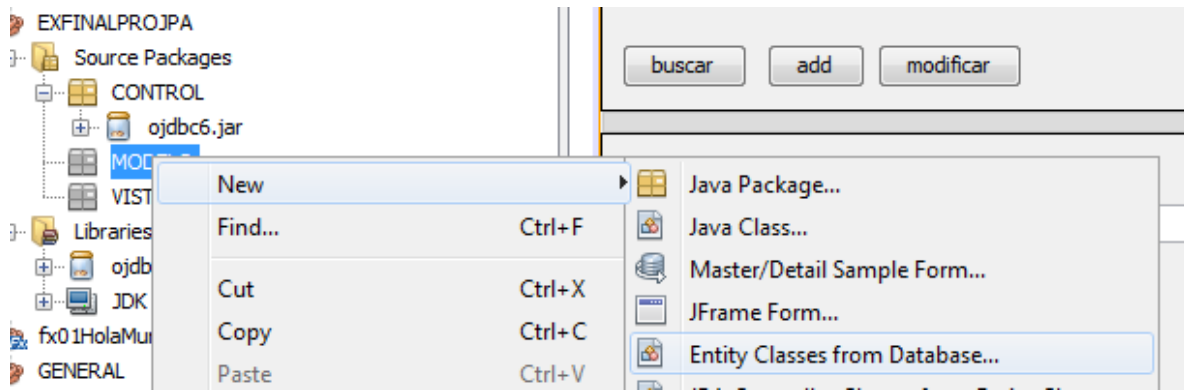


De click derecho sobre Libraries y elija Add JAR/Folder

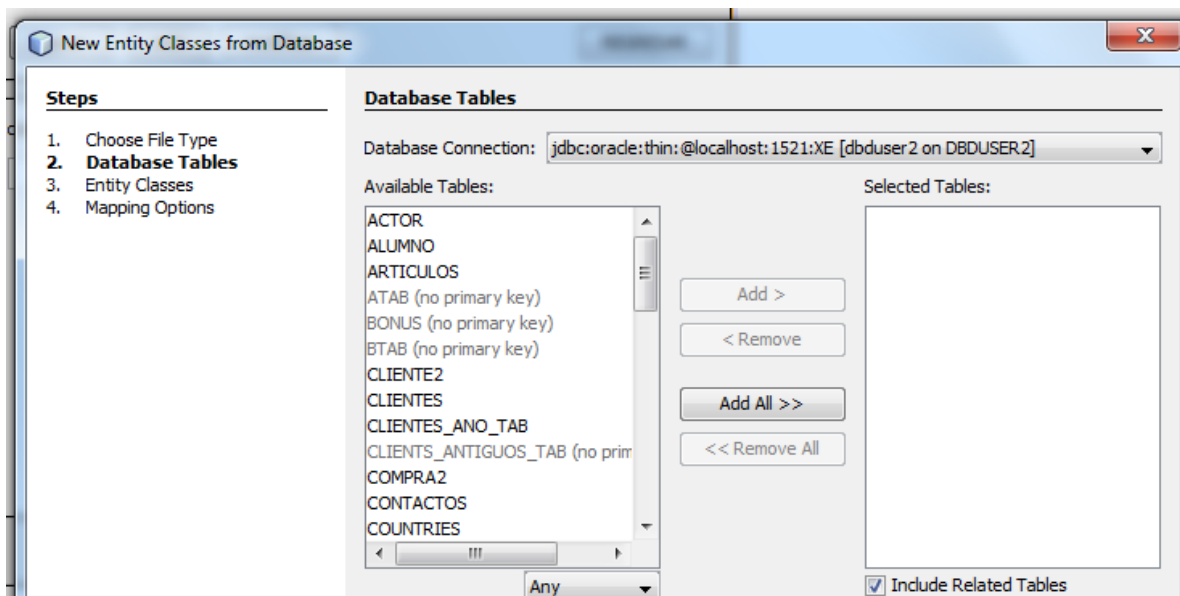
Y busque el archivo jar y adiciónelo.

PASO 2: cree las entidades del modelo

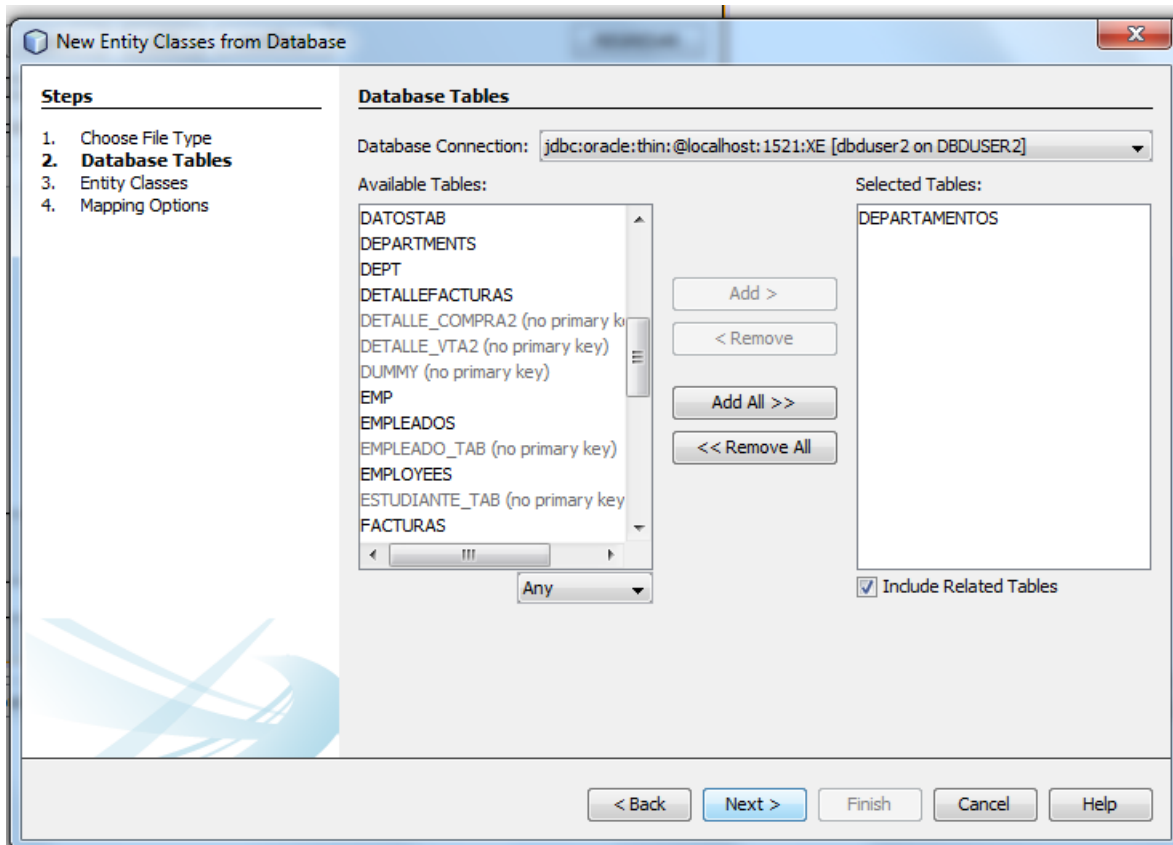
De click derecho sobre MODELO y elija "Entity Classes from database"



Elija la conexión de Oracle si no esta debe crearla

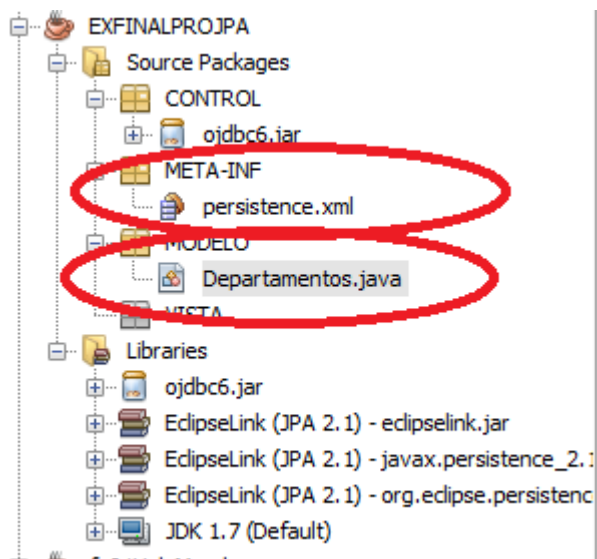


Elija la Tablas Departamentos y presione next



Presione next /next y finish sin cambiar las opciones

El proyecto aparece ahora así:



Se genero la unidad de persistencia (archivo xml) y la entidad (entity) Departamentos

Los cuales tienen esta presentación

Persistence Units

EXFINALPROJPAPU

General:

Persistence Unit Name: EXFINALPROJPAPU

Persistence Library: EclipseLink (JPA 2.1)

JDBC Connection: jdbc:oracle:thin:@localhost:1521:XE [dbduser2 on DBDUSER2]

☐ Use Java Transaction APIs

Table Generation Strategy: ☐ Create ☐ Drop and Create ☒ None

Validation Strategy: ☒ Auto ☐ Callback ☐ None

Shared Cache Mode: ☐ All ☐ None ☐ Enable Selective ☐ Disable Selective ☒ Unspecified

☐ Include All Entity Classes in "EXFINALPROJPA" Module

Include Entity Classes:
MODELO.Departamentos

Y este código XML

Donde el nombre mas imporante esta encerradon en le circulo en rojo **EXFINALPROJPAPU**

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w
3 <persistence-unit name="EXFINALPROJPAPU" transaction-type="RESOURCE_LOCAL">
4 <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
5 <class>MODELO.Departamentos</class>
6 <properties>
7 <property name="javax.persistence.jdbc.url" value="jdbc:oracle:thin:@localhost:1521:XE"/>
8 <property name="javax.persistence.jdbc.password" value="dbduser2"/>
9 <property name="javax.persistence.jdbc.driver" value="oracle.jdbc.OracleDriver"/>
10 <property name="javax.persistence.jdbc.user" value="dbduser2"/>
11 </properties>
12 </persistence-unit>
13 </persistence>
14

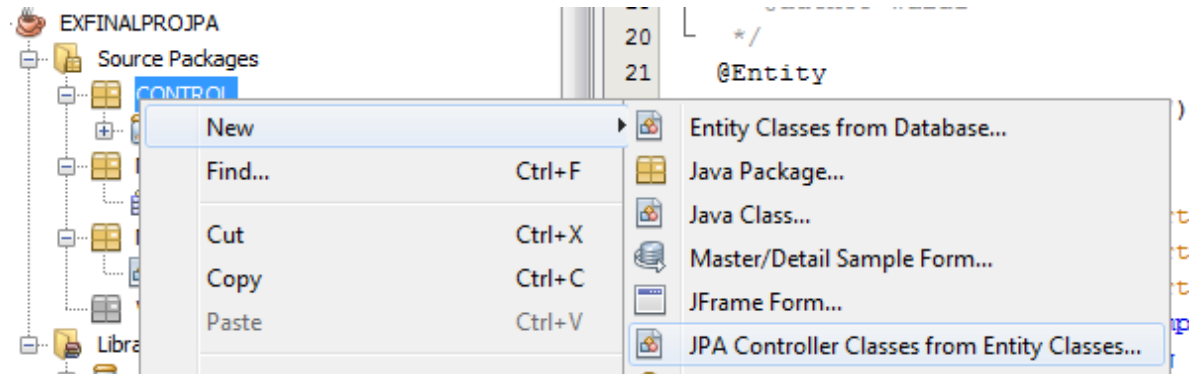
```

PASO 3: CREAR EL CONTROLLER

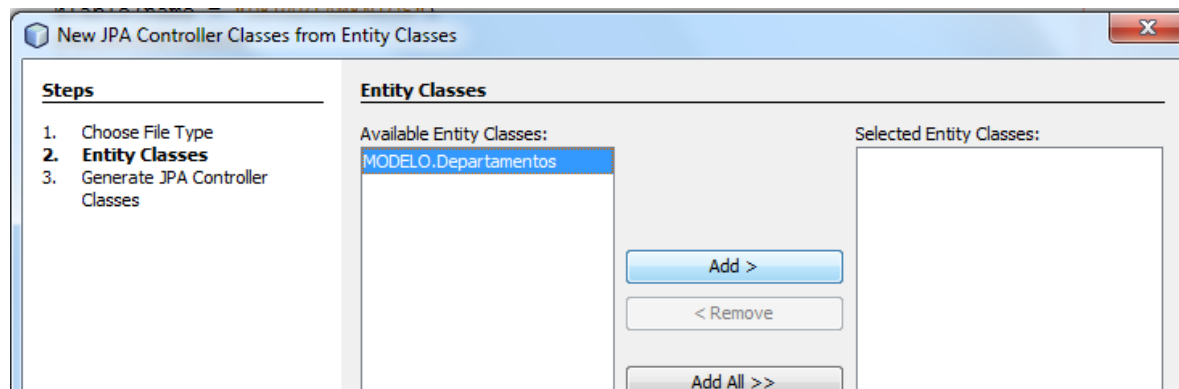
DepartamentosJpaController

Sobre **CONTROL** de Boton derecho del mouse y de new y elija

JPA Controller Classes from Entity Classes como se muestra

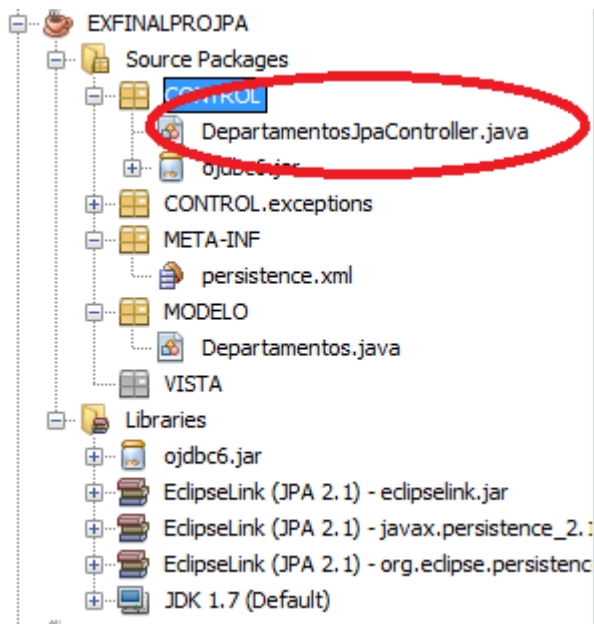


Elija MODELO.Departamentos



Y presione next y luego Finish

Se genera



Y en esta clase se tienen todos estos métodos

```
public void create
```

```
public void edit
```

```
public void destroy
```

```
public List<Departamentos> findDepartamentosEntities()
```

```
public List<Departamentos> findDepartamentosEntities(int maxResults, int firstResult)
```

```
private List<Departamentos> findDepartamentosEntities(boolean all, int maxResults, int firstResult)
```

```
public Departamentos findDepartamentos(Integer id) {
```

```
public int getDepartamentosCount
```

Que permiten crear, modificar, eliminar departemntos y hacer algunas consutlas de información

Lo importante aquí es que las instrucciones para ahcer esto ya están incluidas y todos los procesos con las base de datos

HAGAMOS UNA PRUEBA

Haga Un programa (class) PruebaJPA1

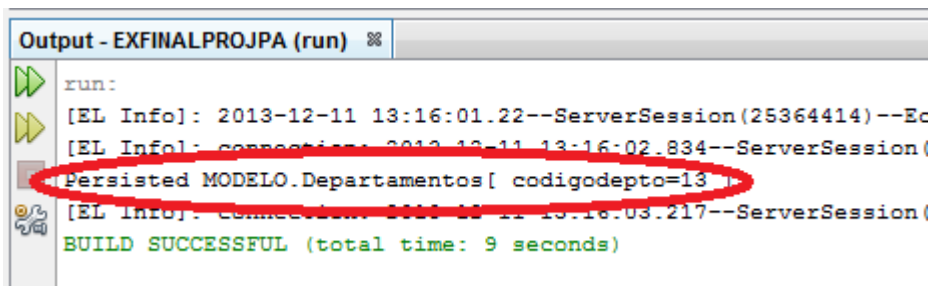
Y meta este main

```
public static void main(String[] a) throws Exception {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("EXFINALPROJPAU");
    EntityManager em = emf.createEntityManager();
    DepartamentosJpaController service = new DepartamentosJpaController (emf);
    em.getTransaction().begin();

    Integer codigodepto=13;
    String nombredepto="RRHH";
    Departamentos dep = new Departamentos();
    dep.setCodigodepto(codigodepto);
    dep.setNombredepto(nombredepto);
    service.create(dep);
    em.getTransaction().commit();
    System.out.println("Persisted " + dep);
    em.close();
    emf.close();
}
```

Ejecutele

Debe salir



```
Output - EXFINALPROJPA (run) %
run:
[EL Info]: 2013-12-11 13:16:01.22--ServerSession(25364414)--Ec
[EL Info]: connection: 2013-12-11 13:16:02.834--ServerSession(
Persisted MODELO.Departamentos[ codigodepto=13 ]
[EL Info]: connection: 2013-12-11 13:16:03.217--ServerSession(
BUILD SUCCESSFUL (total time: 9 seconds)
```

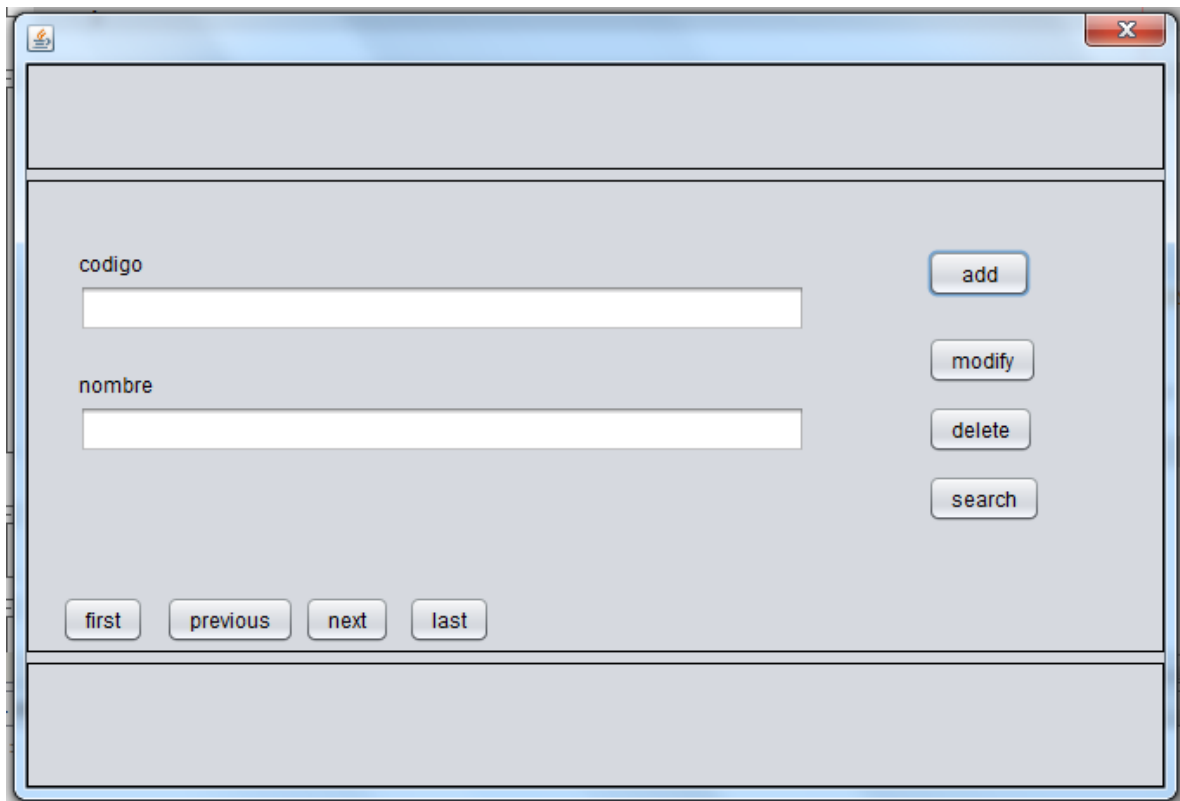
Y en la base de datos debe quedar creado el Departamento 13 RRHH


```
SQL> SELECT * FROM DEPARTAMENTOS;  
CODIGODEPTO NOMBREDEPTO  
-----  
10 sistemas  
11 finanzas  
13 RRHH  
SQL> _
```

Bueno ya estamos listos para trabajar una interface Visual GUI para Departamentos

PASO 1 CREAR LA VENTANA GUI

Cree esta ventana con un JDialog



PASO 2 CODIGO PARA LOS BOTONES

Para poner funcionalidad (lógica) al botón add

Dele doble click al botón add

Y digite este código

```
String scodigo="";
String nombre = "";
scodigo = jTextField1.getText();
nombre = jTextField2.getText();
if(DepartamentosTrBD.adicionar(scodigo, nombre)){
    javax.swing.JOptionPane.showMessageDialog(this, "REGISTRO ADICIONADO");
}
else{
    javax.swing.JOptionPane.showMessageDialog(this, "NO SE PUDO ADICIONAR");
}
```

Observe la clase resaltada no ha sido creada ni por JPA ni por las clases o código que hemos introducido por tanto hay que crearla . esto se hace para que la ventana quede mas limpia de código y las operaciones sean realizadas por clases en el Control

La clase **DepartamentosTrBD** se crea (en el package CONTROL) y se digita el código para el método adicionar así:

```
import MODELO.Departamentos;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author wdiaz
 */
public class DepartamentosTrBD {
    public static boolean adicionar(String scodigo, String nombre) {
        boolean r=true;
        int icodigo=Integer.parseInt(scodigo);
        Integer codigo =icodigo;
        Departamentos dep= new Departamentos();
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("EXFINALPROJPAPU");
        EntityManager em = emf.createEntityManager();
        DepartamentosJpaController service = new DepartamentosJpaController (emf);
```

```

em.getTransaction().begin();
dep.setCodigodepto(codigo);
dep.setNombredepto(nombre);
try{
service.create(dep);
em.getTransaction().commit();

}catch(Exception e){
    System.out.println(e);
    em.getTransaction().rollback();
    r=false;
    return r;
}

System.out.println("Persisted " + dep);
em.close();
emf.close();
return r;
}

```

PASO 3 : verifique que se guardo la Información

Puede probar ejecutar adicionando un nuevo departamento . Haremos el código para buscar mas adelante pero puede verificar si lo adiciono con `SELECT * FROM DEPARTAMENTOS` en Oracle

De igual manera incluya para cada botón este código

PASO 4 CODIGO para los otros Botones

Boton modify

```

String scodigo="";
String nombre = "";
scodigo = jTextField1.getText();
nombre = jTextField2.getText();
if(DepartamentosTrBD.modificar(scodigo, nombre)){
    javax.swing.JOptionPane.showMessageDialog(this, "REGISTRO MODIFICADO");
}
else{
    javax.swing.JOptionPane.showMessageDialog(this, "NO SE PUDO MODIFICAR");
}

```

Boton delete

```
String scodigo="";

scodigo = jTextField1.getText();

if(DepartamentosTrBD.eliminar(scodigo)){
    javax.swing.JOptionPane.showMessageDialog(this, "REGISTRO ELIMINADO CON EXITO");
}
else{
    javax.swing.JOptionPane.showMessageDialog(this, "NO SE PUDO ELIMINAR");
}
```

Boton buscar

```
String scodigo="";

scodigo = jTextField1.getText();
int icodigo=Integer.parseInt(scodigo);
Departamentos dep=DepartamentosTrBD.getDepartamento(icodigo);
if(dep!=null){
    jTextField2.setText(dep.getNombredepto());
}
else{
    javax.swing.JOptionPane.showMessageDialog(this, "NO SE PUDO ENCONTRAR");
}
```

Y la clase **DepartamentosTrBD** completa queda (en le package CONTROL)

```
import MODELO.Departamentos;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author wdiaz
 */
public class DepartamentosTrBD {
    public static boolean adicionar(String scodigo, String nombre) {
        boolean r=true;
        int icodigo=Integer.parseInt(scodigo);
```

```

Integer codigo = icodigo;
Departamentos dep = new Departamentos();
EntityManagerFactory emf = Persistence.createEntityManagerFactory("EXFINALPROJPAPU");
EntityManager em = emf.createEntityManager();
DepartamentosJpaController service = new DepartamentosJpaController (emf);
em.getTransaction().begin();
dep.setCodigodepto(codigo);
dep.setNombredepto(nombre);
try{
    service.create(dep);
    em.getTransaction().commit();

    }catch(Exception e){
        System.out.println(e);
        em.getTransaction().rollback();
        r=false;
        return r;
    }

    System.out.println("Persisted " + dep);
    em.close();
    emf.close();
    return r;
}

public static boolean modificar(String scodigo, String nombre) {
    boolean r=true;
    int icodigo=Integer.parseInt(scodigo);
    Integer codigo = icodigo;
    Departamentos dep = new Departamentos();
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("EXFINALPROJPAPU");
    EntityManager em = emf.createEntityManager();
    DepartamentosJpaController service = new DepartamentosJpaController (emf);
    em.getTransaction().begin();
    dep.setCodigodepto(codigo);
    dep.setNombredepto(nombre);
    try{
        service.edit(dep);
        em.getTransaction().commit();

    }catch(Exception e){
        System.out.println(e);
        em.getTransaction().rollback();
        r=false;
        return r;
    }

    System.out.println("modificado " + dep);
    em.close();

```

```

        emf.close();
        return r;
    }
    public static boolean eliminar(String scodigo) {
        boolean r=true;
        int icodigo=Integer.parseInt(scodigo);
        Integer codigo =icodigo;

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("EXFINALPROJPAPU");
        EntityManager em = emf.createEntityManager();
        DepartamentosJpaController service = new DepartamentosJpaController (emf);
        em.getTransaction().begin();

        try{
            service.destroy(codigo);
            em.getTransaction().commit();

        }catch(Exception e){
            System.out.println(e);
            em.getTransaction().rollback();
            r=false;
            return r;
        }

        System.out.println("modificado " + codigo);
        em.close();
        emf.close();
        return r;
    }
    public static Departamentos getDepartamento(int icodigo){
        Integer codigo=icodigo;
        Departamentos dep= new Departamentos();
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("EXFINALPROJPAPU");
        EntityManager em = emf.createEntityManager();
        DepartamentosJpaController service = new DepartamentosJpaController (emf);

        dep = em.find(Departamentos.class, codigo);
        em.close();
        emf.close();
        return dep;
    }
}

```

Pregunta será posible hacer la lógica (código) para los botones first, previous, next y last?

Algunas ideas al respecto

SEGUNDA PARTE

En la segunda parte se hará el CRUD para la tabla EMPLEADOS teniendo en cuenta que existe una relación 1-n entre Departamentos y Empleados.

ANEXO 1 : SCRIP SQL para crear la base de datos

```
DROP TABLE EMPLEADOS;
DROP TABLE DEPARTAMENTOS;
CREATE TABLE departamentos (
  codigodepto NUMBER(5) NOT NULL,
  NOMBREDEPTO VARCHAR2(25) NOT NULL,
  CONSTRAINT PK_departamentos PRIMARY KEY (codigodepto)
)
;

CREATE TABLE EMPLEADOS (
  CODIGOEMP NUMBER(5) NOT NULL,
  APELLIDO VARCHAR2(40) NOT NULL,
  NOMBRE VARCHAR2(40) NOT NULL,
  SEXO CHAR(1) NOT NULL,
  EMAIL VARCHAR2(80) NOT NULL,
  SALARIO NUMBER(12,2) NOT NULL,
  codigodepto NUMBER(5) NOT NULL,
  CONSTRAINT PK_EMPLEADOS PRIMARY KEY (CODIGOEMP)
)
;

ALTER TABLE EMPLEADOS
  ADD CONSTRAINT FK_EMPLEADOS_departamentos
  FOREIGN KEY (codigodepto) REFERENCES departamentos (codigodepto)
;
```

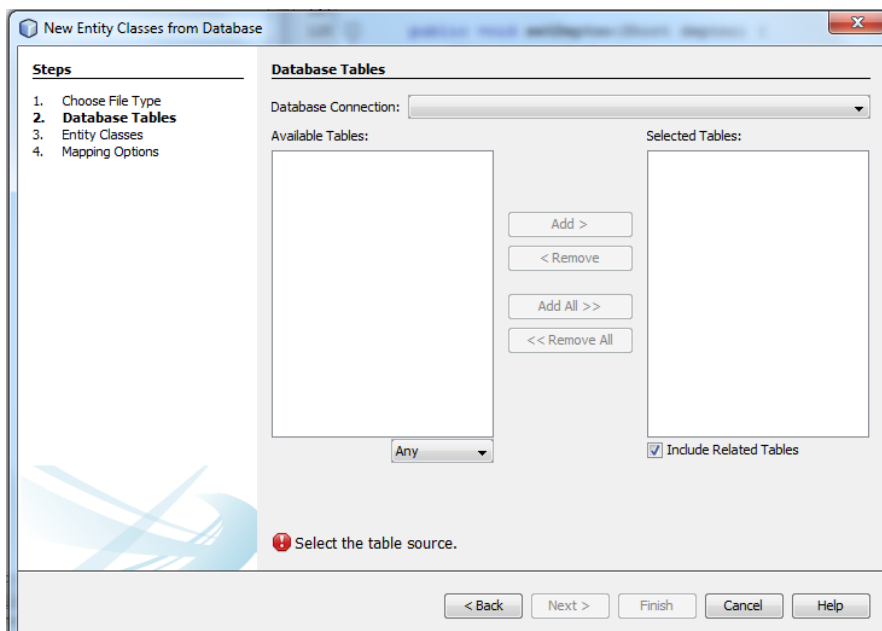
Se aconseja hacer algunos insert para tener dos o tres registros en cada tabla

```
INSERT INTO departamentos VALUES(10,'sistemas');
INSERT INTO departamentos VALUES(11,'finanzas');
INSERT INTO EMPLEADOS
VALUES(1,'GRISALES','AMPARO','F','amparito@gmail.com',5000.0,11);
```

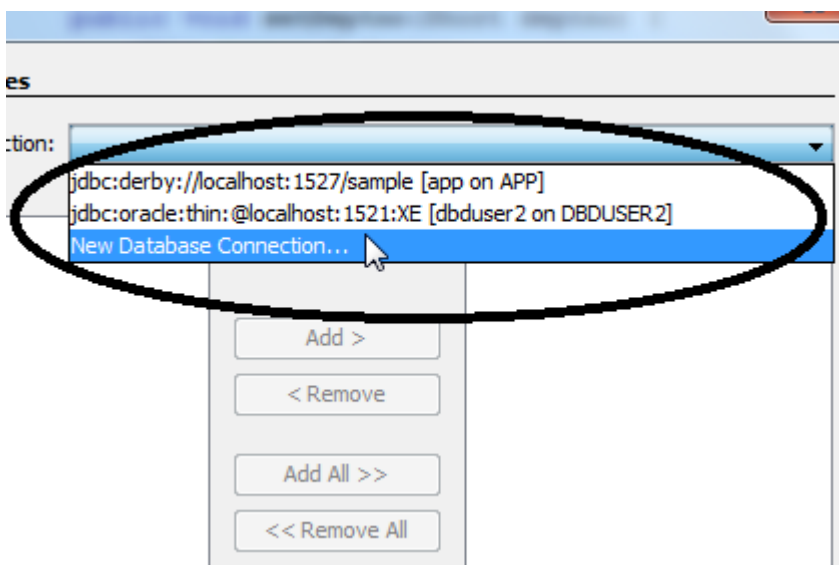
ANEXO 2 EN CASO QUE DEBA CREAR LA CONNECCION DESDE CEROS

Al elegir esta opción se muestra este dialogo para crar una conexión con la base de datos

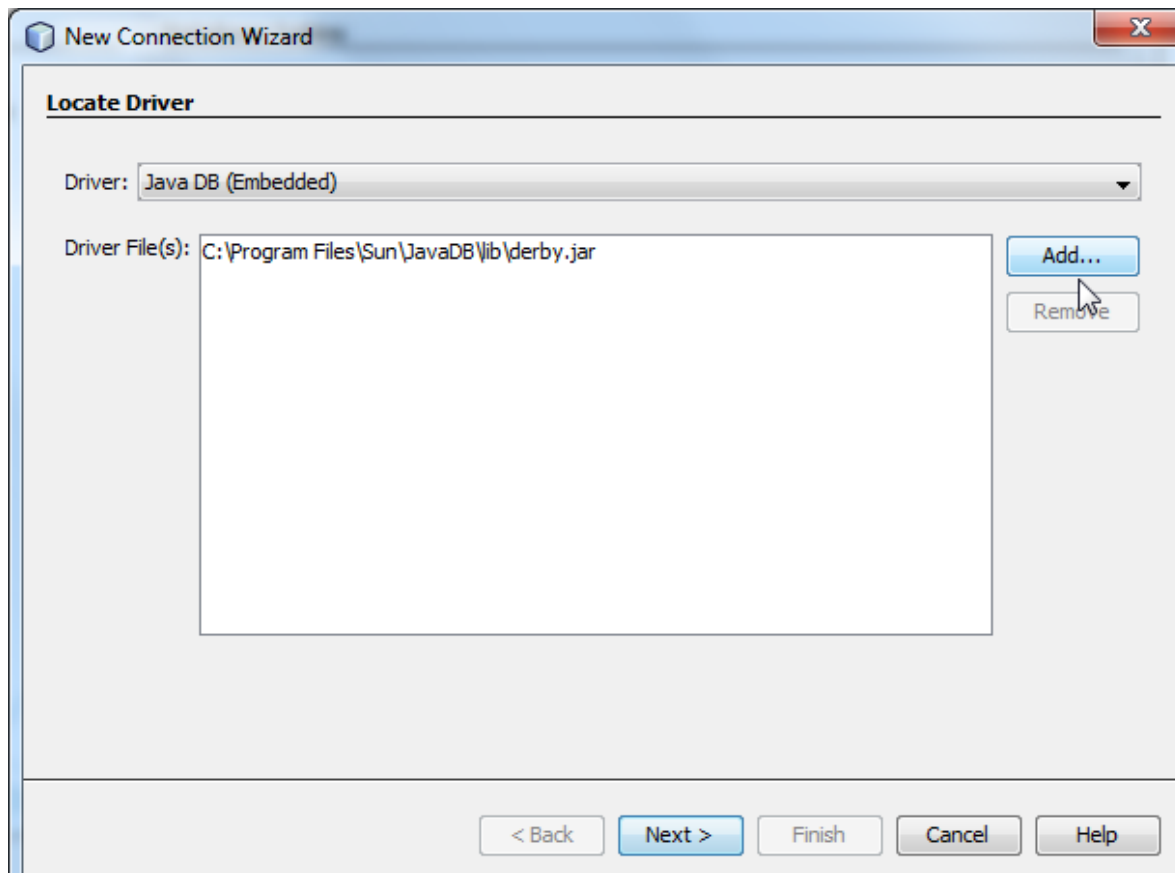
Si la coneccion no esta usted puede crearla eligiendo NEW DATABASE CONECCTION



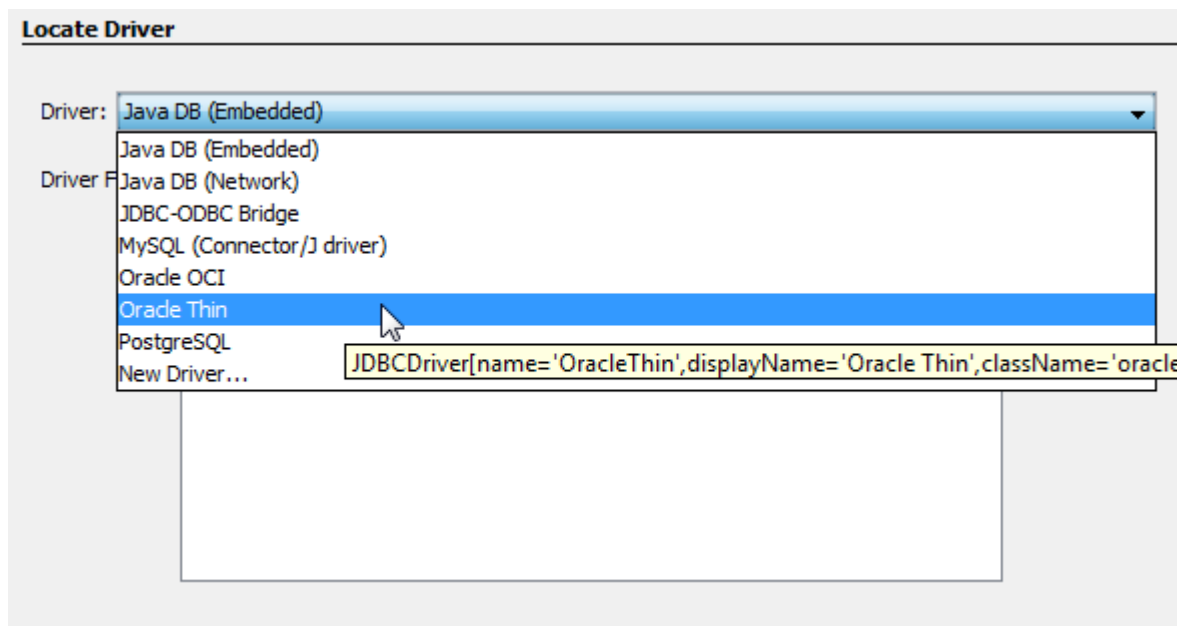
ELIJA NEW DATABASE CONNECTION



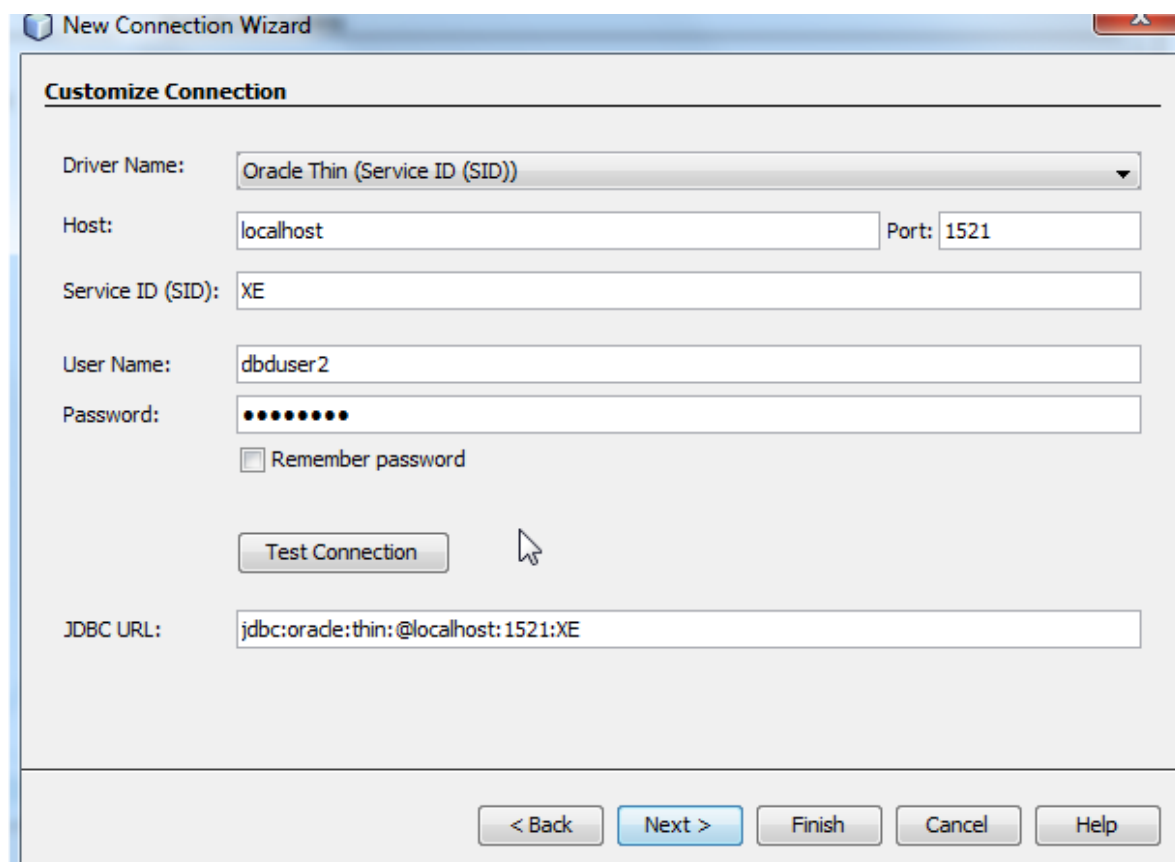
LUEGO ELIJA Oracle Thin en la caja desplegable de drivers



Y ESCOJA



Digite luego el User Name y el password y presione Test Connection



Y si todo esta bien presiones Finish